VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Rozšíření informačního systému pro firmu Novatech

Extension of the Information System for Novatech Company

2010 Bc. Ján Rebroš

Súhlasím so zverejnením tejto diplomovej práce podľa požiadavkov čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava*.

V Ostrave 7.5.2010 .....

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

V Ostrave 7.5.2010 .....

Rád by som na tomto mieste poďakoval vedúcemu diplomovej práce Ing. Marekovi Běhálkovi Ph.D. za odborné vedenie práce, cenné rady a pripomienky.

Abstrakt

Cieľom tejto práce bolo navrhnúť a realizovať rozšírenie informačného systému Inside pre firmu Novatech. Rozšírenie umožnilo prepojenie systému s ERP informačným systémom Money S5 a webovou stránkou firmy. Práca ozrejmuje súčasný stav vo firme z hľadiska využívania systémov, ich funkcionality, architektúry a použitých technológií. Popisuje základne pojmy z oblastí technológií využívaných ku komunikácií medzi aplikáciami a možnosti vzájomnej výmeny informácií. Následne približuje postup špecifikácie požiadaviek zadávateľa, analýzu týchto požiadaviek a návrh riešenia v zvolenej architektúre. V poslednej časti práce sú praktické ukážky zaujímavých riešení zo samotnej implementácie rozšírenia.

Kľúčové slová: Informačný systém, Biznis procesy, XML, Webové služby, Replikácia

Abstract

The main objective of this project is to design and produce extension of information system for Novatech company. The project interconnects ERP information system Money S5 and a web page of the company. Firstly, I am introducing already used technologies in Novatech, mainly its systems in view of functionality and architecture. In the next part, I am explaining basic terms of technologies that are used in communication between applications. The next step is to show how to specify requirements, analysis and design of the architecture of the system. In the final part I am presenting interesting parts of the implemented system.

Keywords: Information system, Business processes, XML, Web services, Replication

Zoznam použitých skratiek a symbolov

AJAX	–	Asynchronous JavaScript and XML
API	–	Application programming interface
ERP	–	Enterprise resource planning
DOM	–	Document Object Model
DTD	–	Document Type Definition
FTP	–	File Transfer Protocol
HTML	–	Hypertext Markup Language
HTTP	–	Hypertext Transfer Protocol
JSON	–	JavaScript Object Notation
MVC	–	Model-View-Controller
PHP	–	Hypertext Preprocessor
RELAX NG	–	Regular Language for XML Next Generation
REST	–	Representational State Transfer
RIA	–	Rich Internet Application
RPC	–	Remote procedure call
RSS	–	Really Simple Syndication
SAX	–	Simple API for XML
SMTP	–	Simple Mail Transfer Protocol
SOAP	–	SIMPLE Object Access Protocol
SQL	–	Structured Query Language
SVG	–	Scalable Vector Graphics
UDDI	–	Universal Description, Discovery and Integration
URI	–	Uniform Resource Identifier
URL	–	Uniform Resource Locator
W3C	–	World Wide Web Consortium
WS	–	Web Services
WSDL	–	Web Services Description Language
XAML	–	Extensible Application Markup Language
XHTML	–	Extensible Hypertext Markup Language
XML	–	Extensible Markup Language
XSL	–	Extensible Stylesheet Language
XSLT	–	Extensible Stylesheet Language Transformation

Obsah

Zoznam obrázkov

Zoznam výpisov zdrojového kódu

1 Úvod

V dnešnej dobe sa čoraz viac kladie dôraz na komunikáciu. A to nie len medzi ľuďmi, ale aj medzi aplikáciami. Potrebou moderných firiem je, aby ich informačné systémy umožnili komunikáciu s inými systémami, kvôli vzájomnej výmene informácií alebo replikácii dát.

Táto práca popisuje rozšírenie informačného systému Inside používaného servisným oddelením firmy Novatech, ktorý bol implementovaný v rámci mojej bakalárskej práce. Rozšírenie je zamerané na prepojenie systému s ERP informačným systémom Money S5 a webovou stránkou firmy. Popisuje priebeh špecifikácie, analýzy, návrhu a samotnej implementácie rozšírenia.

V úvode práce je venovaný súčasnému stavu vo firme Novatech z hľadiska využívania informačných systémov. Pozornosť je pri tom venovaná funkcionalite a architektúre systému Inside. Vysvetlené je použitie viacvrstvovej architektúry pomocou Model-View-Controller. Taktiež nechýba stručný opis systému Money, jeho možnosti využitia, použité technológie a architektúra.

Ďalšia kapitola rozoberá možnosti komunikácie medzi aplikáciami. Opísaný je formát XML a s ním súvisiace technológie, ako sú schémy, transformácie, či jeho spracovanie. Následne je objasnené fungovanie webových služieb využívajúcich XML. Vysvetlené sú pojmy ako SOAP, WSDL, REST, či AJAX.

V štvrtej kapitole sú upresnené špecifikácie zadania. Špecifikáciou biznis modelov sú opísané procesy vo firme, ktoré by prepojením systémov mohli byť riešene oveľa efektívnejšie. Na ich základe sú zhrnuté požadované funkcie a vlastnosti, ktoré je potrebné v rámci rozšírenia implementovať.

Piata kapitola obsahuje špecifikáciu požiadaviek. Prostredníctvom funkčnej špecifikácie opisuje systém z pohľadu nových funkcií, ktoré má vykonávať a pomocou špecifikácie chovania sú zachytené ich dynamické vlastnosti.

Šiesta kapitola Analýza skúma špecifikované požiadavky z pohľadu objektov, ktoré je možné nájsť v riešenej doméne. Definuje vzájomnú spoluprácu medzi nájdenými objektmi, ktorá vedie k splneniu ich funkcionality. Objekty popisuje a štrukturalizuje do podoby triedneho diagramu.

Siedma kapitola Návrh popisuje proces vytvárania modelu návrhu na základe modelu analýzy. Upresňuje použitie modelu analýzy v skutočnom implementačnom prostredí, teda použitie v architektúre systému. Upresnený je tiež spôsob replikácie dát. Modelom nasadenia je špecifikovaná fyzická štruktúra prevádzkových prostriedkov.

V ôsmej kapitole sú vysvetlené a opísané niektoré zaujímave riešenia a technológie použité pri implementácií rozšírenia systému. Záver kapitoly tvorí zhrnutie problematiky fungovania komunikácie medzi systémami v praxi.

2 Súčasný stav

Firma Novatech je súkromná spoločnosť orientovaná na špecializovaný predaj výpočtovej, digitálnej a kancelárskej techniky pre segment malých, stredne veľkých firiem a domácich používateľov. Vysoký dôraz kladie na implementačné a podporné služby pre zákazníkov.

V súčastnosti k svojej činnosti využíva okrem iného dva informačné systémy. Money S5 ako ERP informačný systém, ktorý rieši vo firme účtovníctvo, obchod, sklady a veci s tým súvisiace. Druhým systémom je vyvinutá webová aplikácia Inside, ktorá slúži servisnému oddeleniu firmy na evidovanie servisných zákaziek. Tieto systémy momentálne fungujú úplne samostatne a jediným možným prepojením sú ich používatelia prepisujúci dáta.

2.1 Informačný systém Inside

Inside je informačný systém firmy Novatech, využívaný predovšetkým servisným oddelením. Umožňuje evidovať a spracovávať kompletnú agendu servisných zákaziek. Servisné zákazky môžu byť rôzneho typu, ako napríklad reklamácia, servis, výjazd. Počas riešenia prechádzajú rôznymi stavmi. K zákazkám podľa ich typu a stavu aplikácia umožňuje tlač potrebných formulárov a iných dokumentov. V systéme sú tiež evidovaný zákazníci a partneri servisného oddelenia. Informácie o zákazníkovi je potrebné evidovať pri každej zákazke. Pri prijímaní novej zákazky môže byť v minulosti uložený zákazník vybraný zo zoznamu zákazníkov. Zoznam partnerov slúži pri preposielaní zákazky do iného servisného oddelenia. Popri samotných údajoch o zákazke eviduje systém aj mnohé pomocné údaje, ako je napríklad história stavov zákazky - kto, kedy a čo so zákazkou vykonal, správy predávané medzi používateľmi o zákazke, cenník služieb, dohodnuté termíny. Z informácií o zákazke systém generuje niektoré zaujímavé a pre firmu potrebné štatistiky. Používatelia systému majú podľa zaradenia v skupine pridelené rôzne práva na vykonanie funkcií a zobrazenie informácií.

Systém bol navrhnutý a implementovaný v rámci mojej bakalárskej práce [8]. Medzičasom bol ďalej vyvíjaný a doplňovaný o nové funkcie. Na základe požiadaviek zadávateľa bol systém naprogramovaný ako webová aplikácia s použitím voľne dostupných technológií. Preto je aplikácia vytvorená v PHP s použitím šablónovacieho systému Smarty a s využitím architektúry Model-View-Controller (MVC). Ako úložisko dát je používaná MySQL databáza.

Webová aplikácia je umiestnená na serveri, odkiaľ je prístupná ktorémukoľvek zariadeniu (klientovi) s internetovým prehliadačom a pripojením na internet. Na klienta už nie je potrebné nič inštalovať, preto je aplikácia prístupná z rôznych operačných systémov a zariadení, ako napríklad aj z mobilného telefónu.

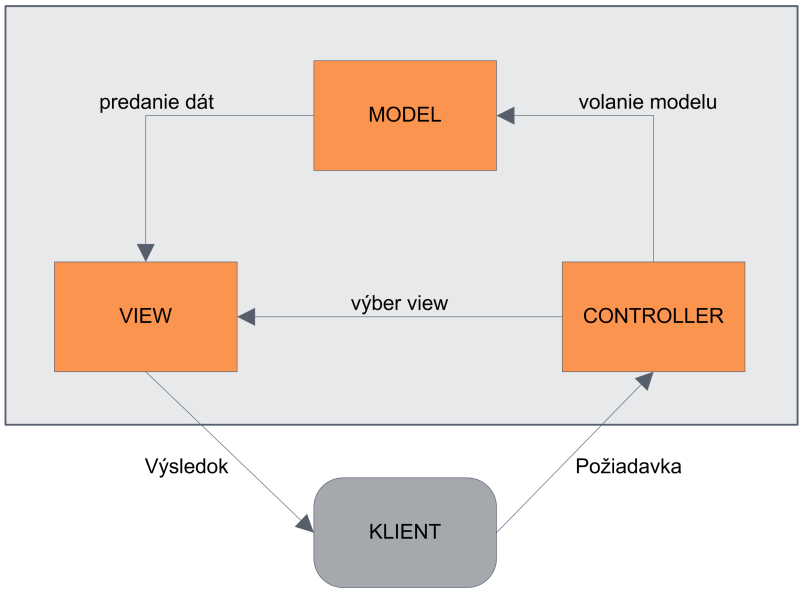
2.2 Architektúra aplikácie

Architektúra aplikácie je konštruovaná ako viacvrstvová (multi-layer), minimálne môžme hovoriť o troch vrstvách - prezentačnej, aplikačnej a dátovej. Návrh aplikácie vo

viacvrstvovej architektúre prináša mnoho výhod, napríklad že modifikácia niektorého z komponentov nevyvolá zmeny v ďalšom prvku ostatných vrstiev. To platí i pre prípadnú migráciu úložiska dát do iného databázového systému. Nesmieme tiež zabudnúť na väčšiu prehľadnosť zdrojových kódov a tým aj možné oveľa jednoduchšie budúce rozšírenie, prípadne priebežnú modifikáciu.

2.2.1 Model-View-Controller

MVC je obecná koncepcia objektovej architektúry, ktorá rieši základné oddelenie vyššie uvedených vrstiev. Ako je vidieť na obrázku 1, MVC rozdeľuje aplikáciu na tri logické časti: Model, View a Controller. Pre každú z nich definuje za čo je v rámci aplikácie zodpovedná.



Obr. 1: Model-View-Controller

Model je časť aplikácie, ktorá vykonáva všetku aplikačnú logiku. Je funkčným a dátovým základom celej aplikácie. Poskytuje prostriedky pre prístup k dátovej základni a stavom aplikácie, ako aj pre ich ukladanie a aktualizáciu. Ako celok je zapuzdrený a pre view a controller poskytuje presne definované rozhranie. Pri používaní perzistentného úložiska dát (databázy), je potrebné riešiť otázku vzťahu medzi aplikačnou vrstvou a dátovou vrstvou, ktorá nie je architektúrou MVC pokrytá.

V systéme predstavujú jednotlivé modely triedy, ktoré sú definované v triednom diagrame. Model vykoná potrebné operácie s dátami a prípadne predá dáta prostredníctvom šablónovacieho systému view. Napríklad pri pridávaní novej zákazky, model skontroluje a uloží, alebo inak spracuje používateľom vložené dáta z formulára do da-

tabázy. Naopak pri detaile zákazky, keď je potrebné údaje zobraziť, tieto dáta z databázy vyberie, môže ich ešte prípadne spracovať a následne ich predá view na zobrazenie.

View zobrazuje obsah modelu, zaisťuje grafický či iný výstup aplikácie. Pristupuje k dátam aplikácie cez model a špecifikuje ako majú byť prezentované. Pri zmene stavu modelu sa aktualizuje aj zobrazenie. Pri webových aplikáciách zvyčajne view generuje HTML kód, ktorý je odoslaný do prehliadača, ako odpoveď na požiadavku. Je možné mať jeden Model a niekoľko rôznych view a teda rovnaké dáta prezentovať v rôznych formátoch.

V systéme predstavujú view jednotlivé šablóny stránok, ktoré sú pomocou šablónovacieho systému spracované a prevedené do konečného HTML kódu, ktorý je prezentovaný používateľovi v prehliadači. Napríklad pre pridanie novej zákazky existuje šablóna s formulárom, ktorý obsahuje všetky potrebné políčka na vyplnenie údajov. Naopak pri detaile zákazky, keď je potrebné údaje zobraziť, sa použije šablóna, kde sú uložené údaje len zobrazované, napríklad v tabuľke.

Ako šablónovací systém v aplikácii je použitý Smarty pre PHP [17]. Od iných šablónovacích systémov sa líši predovšetkým spôsobom akým šablóny môže spracovávať. Pri prvom volaní (tzn. pri prvom spustení skriptu, ktorý pre svoj výstup používa šablónu) sú šablóny prevedené (skompilované) do podoby PHP skriptu. Pri ďalšom volaní je následne spustená skompilovaná verzia šablóny, čo výrazne zrýchľuje odozvu servera. Alternatívou ku kompilovaniu šablón je tzv. cachovanie obsahu. V princípe sa jedná o rovnaký postup, nie sú však vytvárané PHP skripty, ale HTML.

Napriek tomu, že je Smarty určený k oddeleniu aplikačnej a prezentačnej logiky, nie je obmedzený iba na tvorbu šablón pomocou HTML značiek. Umožňuje použitie riadiacich štruktúr, cyklov, vstavaných funkcii na formátovanie reťazcov, dátumov atď. Použitie funkcii v prezentačnej časti nie je obmedzené a to vďaka tzv. pluginom. Niekoľko ich je obsiahnutých v základe, ostatné si je možné jednoducho doprogramovať v PHP.

Controller definuje chovanie aplikácie. Spracováva všetky vstupy a udalosti pochádzajúce od používateľa. Na ich základe vyvolá príslušné procesy modelu, mení jeho stav (resp. dáta). Podľa udalostí prijatých od používateľa vyberie controller vhodné view pre ďalšie zobrazenie. V prípade webových aplikácii sú hlavným vstupom HTTP požiadavky.

V systéme sú všetky požiadavky od klienta pomocou servera presmerované na Front Controller. V ňom pomocou tzv. routrovania sa zistí zo zadanej adresy URL, ktorý konkrétny Action Controller a ktorá akcia (metóda) sa má zavolať. Následne pomocou tzv. dispatchingu sa zavolá príslušný Action Controller a jeho metóda s parametrami. V metóde je potom volaný potrebný model a view.

Komponenty view a controller sú v štandardnom rozdelení vrstiev na prezentačnú, aplikačnú a dátovú obvykle zaraďované ako prezentačná vrstva. V MVC je teda prezentačná vrstva rozdelená na view a controller, napriek tomu najdôležitejšie rozdelenie je medzi prezentačnou a aplikačnou vrstvou.

2.2.2 Dátová vrstva

Dátová vrstva je implementovaná pomocou návrhového vzoru Factory Method, aby bolo možné jednoduchým pridaním ovládača databázového systému a zmenou v konfiguračnom súbore systému určiť aký databázový SQL systém bude aplikácia používať.

Návrhový vzor Factory Method [4] patrí do skupiny tvoriacich vzorov. Slúži k vytvoreniu inštancie jednej z niekoľkých možných tried, často v závislosti na poskytnutých dátach. Obvykle všetky triedy, ktoré vracia, majú rovnakú nad-triedu a metódy, ale každá z nich vykonáva úlohu iným spôsobom a je optimalizovaná pre iný typ dát. Rozhodnutie o tom, ktorú triedu vrátiť necháva tvoriaca trieda na svojej podtriede a sama pracuje nezávisle na zvolenej produktovej triede.

V systéme to vyzerá tak, že trieda *Database*, ktorá má metódu *getInstance()* vráti po jej zavolaní, podľa konfigurácie, vytvorenú inštanciu ovládača vybraného databázového systému (napr. *Database_MySQL*). Ten už obsahuje potrebné metódy na prácu nad databázovým systémom. Inštancia na triedu *Database* sa vytvára v Modeli, ktorý s databázou pracuje. Pri prípadnej zmene databázového systému stačí zmeniť v konfiguračnom súbore aplikácie názov ovládača databázy a samozrejme pridať tento ovládač databázového systému. V ostatných vrstvách aplikácie už nie je potrebné nič meniť.

Pri zmene databázového systému je však potrebné postupovať opatrne, kvôli možným rozdielom v SQL dialekte medzi jednotlivými databázovými systémami a tieto rozdiely buď ošetriť v ovládači, alebo úpravou jednotlivých SQL príkazov.

V aplikácií sú využívané aj vlastnosti a funkcie databázových systémoch ako sú referenčná integrita, transakcie a triggery.

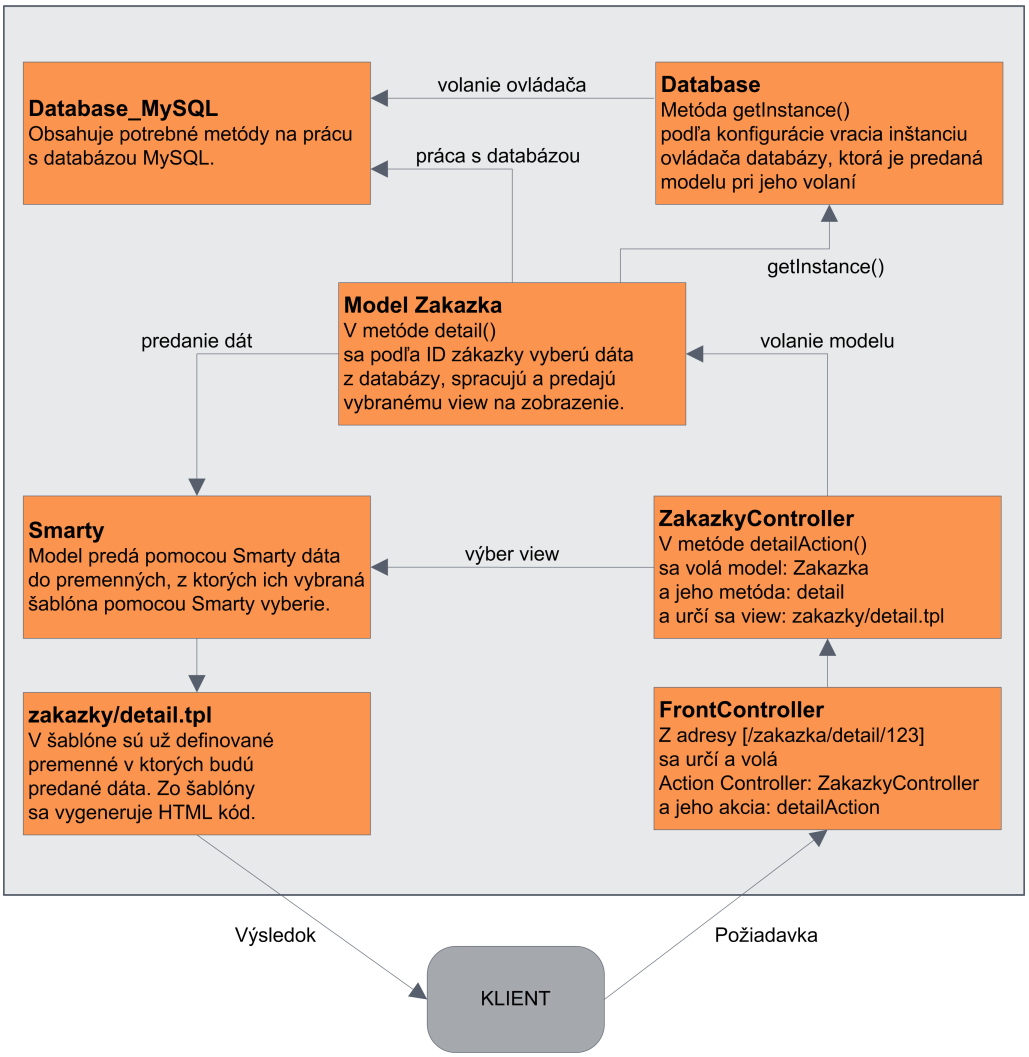
Referenčná integrita je nástroj databázového systému, ktorý pomáha udržiavať vzťahy v relačne prepojených databázových tabuľkách. Referenčná integrita sa definuje *cudzím kľúčom* pre dvojicu tabuliek. Tabuľka, v ktorej je pravidlo uvedené sa nazýva podriadená tabuľka (slave). Tabuľka, ktorej meno je v obmedzení uvedené je nadriadená tabuľka (master). Pravidlo referenčnej integrity vyžaduje, aby každý záznam použitý v podriadenej tabuľke existoval v nadriadenej tabuľke.

Transakčné spracovanie znamená, že skupina logických operácii je chápaná ako transakcia. Transakcia musí byť vždy vykonaná ako jeden celok. Pokiaľ sa pri spracovaní v rámci transakcie vyskytne akákoľvek chyba a transakcia nemôže byť dokončená, všetky čiastkové operácie musia byť vrátené do stavu pred začiatkom transakcie.

Trigger je uložená procedúra, ktorá sa spúšťa v súvislosti s prevedením nejakého akčného dotazu nad tabuľkou. *V súvislosti* znamená, že trigger sa môže spustiť buď predtým, ako je úprava dát vykonaná, alebo potom, čo sú zmeny v dátach zapísané do databázy. *Akčný dotaz* znamená, že trigger je možné spustiť pri vkladaní dát, pri ich aktualizácii alebo pri odstraňovaní dát z databáze. Vo vnútri triggeru je možné mať napríklad cyklus, podmienku, lokálnu premennú, matematický výpočet a podobne.

2.2.3 Kompletná architektúra

Na obrázku 2 je znázornená kompletná architektúra aplikácie, ktorá zahŕňa všetky vrstvy. Zobrazuje spracovanie požiadavky *detail zákazky*, pričom priebeh spracovania je podobný aj u ostatných požiadaviek.

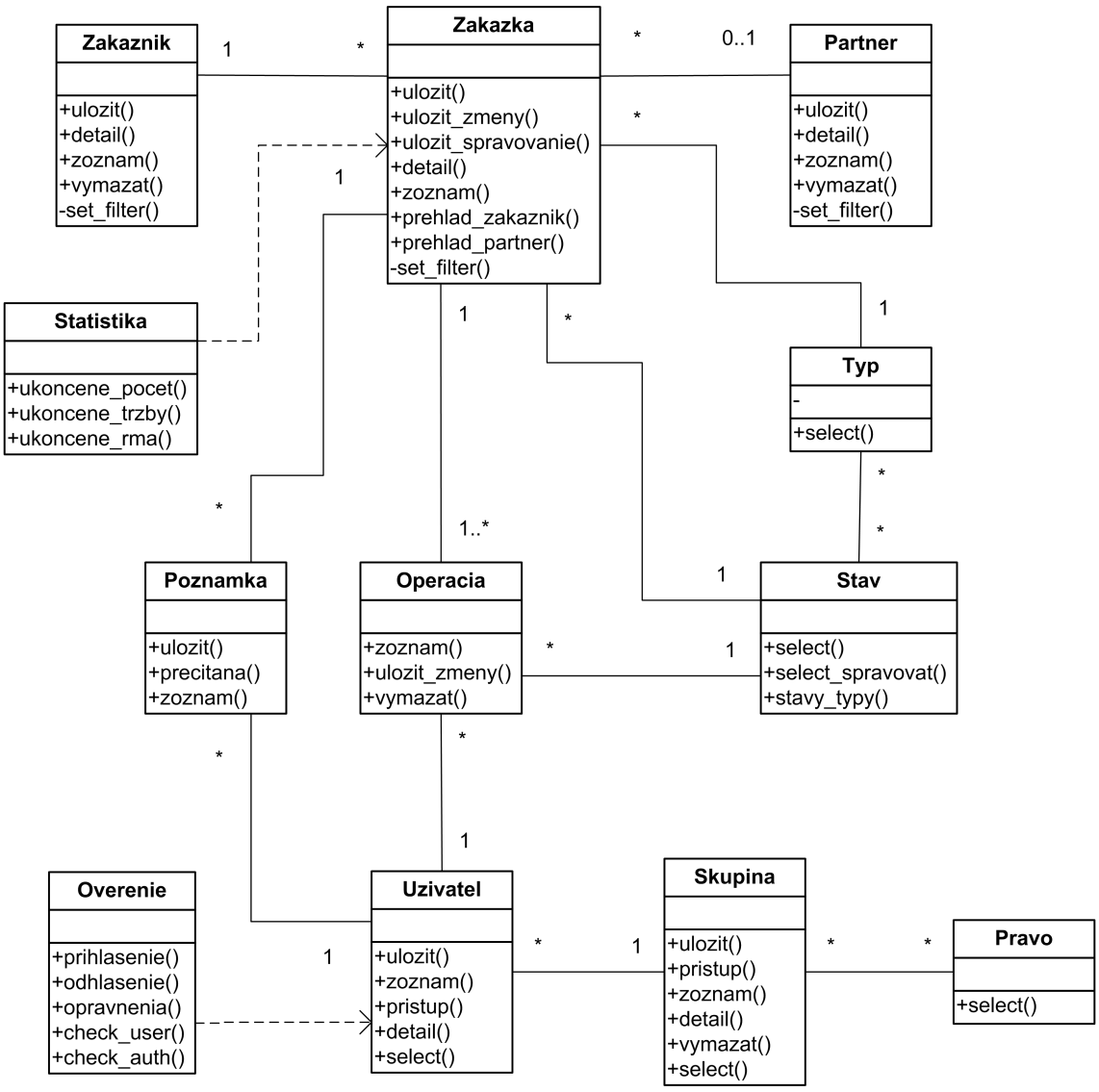


Obr. 2: Kompletná architektúra systému Inside

Klient odošle požiadavku na server. Ten z požiadavky pomocou FrontControllera určí a zavolá príslušný ActionController s potrebnou akciou. ActionController už volá príslušný model a vyberá k nemu potrebné view. Model volá metódu *getInstance()* triedy *Database*, ktorá vracia inštanciu na ovládač databázového systému. Model môže pomocou inštancie pracovať s metódami ovládača a tým pristupovať k databáze. Dáta predá view na spracovanie a tie sú pomocou šablóny vrátené klientovi ako HTML stránka.

2.2.4 Statická štruktúra systému

Statická štruktúra systému špecifikuje jeho prvky (entity), vnútornú štruktúru týchto entít a vzťahy medzi nimi. Statická štruktúra systému je vyjadrená pomocou triedneho diagramu na obrázku 3. V diagrame sú kvôli prehľadnosti znázornené len dôležitejšie súčasti systému. Podrobnejšia špecifikácia aplikácie je rozobraná v bakalárskej práci [8].

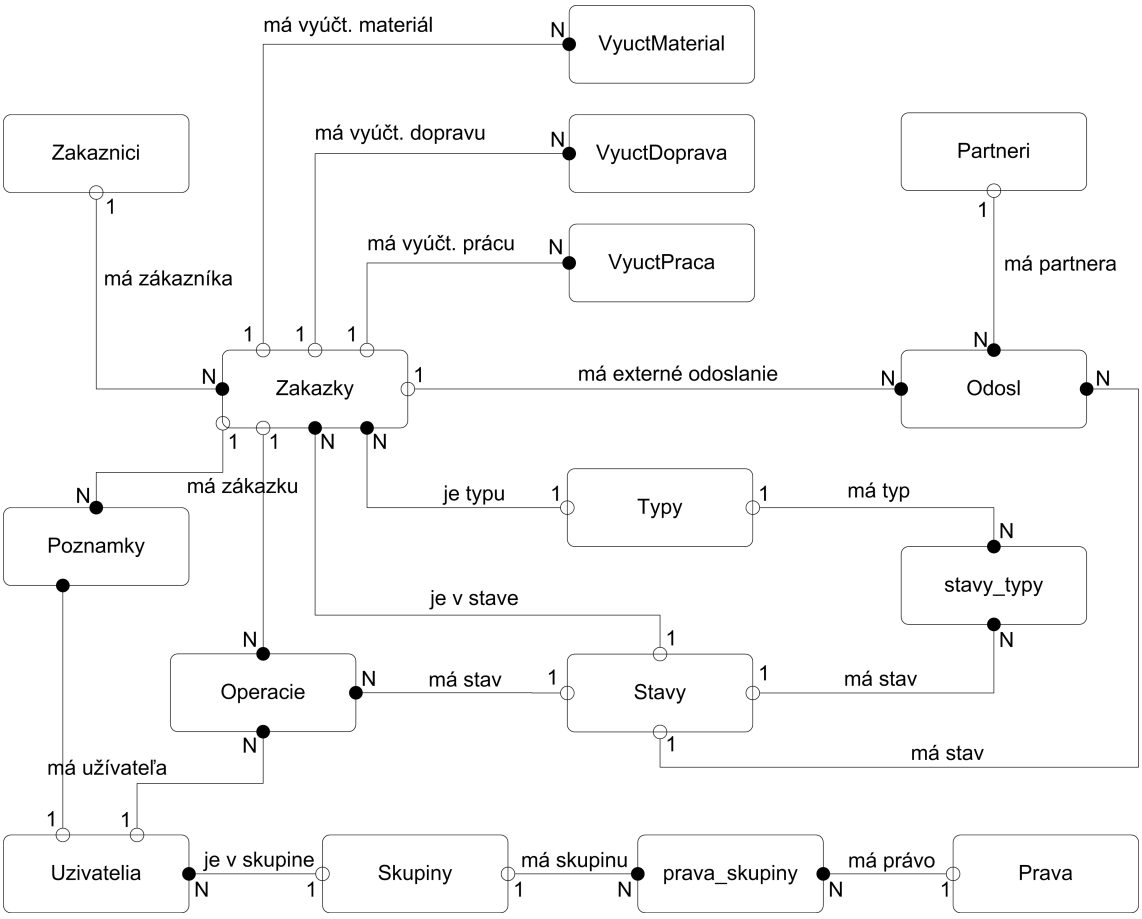


Obr. 3: Triedny diagram systému Inside

2.2.5 Konceptuálny dátový model

Konceptuálny dátový model predstavuje určité zobecnenie, oproti konkrétnej implementácii dátovej štruktúry. Zobecnením sa získava nezávislosť modelu na konkrétnom databázovom systéme, ale zároveň je model možné kedykoľvek previesť do konkrétneho implementačného prostredia. Model umožňuje identifikovať entitné typy a väzby medzi nimi, ktoré sú v systéme evidované.

Jednotlivé triedy systému z triedneho diagramu sú namapované na tabuľky databázy. Asociácia 1:N vedie k definovaniu cudzieho kľúča v detail tabuľke, ktorý odkazuje na príslušný riadok master tabuľky identifikovaný jednoznačne hodnotou primárneho kľúča. Asociácia M:N medzi triedami vedie k vytvoreniu väzobnej tabuľky, ktorá obsahuje cudzie kľúče odkazujúce na obe *master* tabuľky. Špecifikácia atribútov je k dispozícii v bakalárskej práci [8].



Obr. 4: ER diagram systému Inside

2.3 Informačný systém Money S5

Money S5 [16] je moderný podnikový informačný (ERP) systém pre stredné a väčšie spoločnosti od firmy CÍGLER SOFTWARE. Používa modernú viacvrstvovú architektúru - k jedinému serveru sa môžu pripájať desiatky používateľov na klientskych staniciach v pobočkách a inde. Komunikácia sa uskutočňuje cez internú sieť alebo internet, prostredníctvom vzdialenej správy dát z ktoréhokoľvek počítača, mobilného telefónu či prenosného zariadenia. V prípade menších predajní môže byť klient aj server nainštalovaný na jednom počítači.

Funkčnosť aplikácie je z technologických aj používateľských dôvodov rozdelená do modulov podľa jednotlivých oblastí riešenej problematiky. Jednotlivé moduly sa potom môžu pripájať k rôznym inštanciám (čiastkovým súborom dát) databázového servera. Firma Novatech využíva pri svojej činnosti napríklad moduly: účtovníctvo, fakturácia, adresár, sklady, cenníky, objednávky, majetok a ďalšie.

Aplikácia je vytvorená v prostredí Microsoft .NET a v jazyku C#. K svojej prevádzke využíva Microsoft .NET Framework verzie 3.5. K uchovávaniu účtovaných dát a všetkých informácií, ktoré systém spracováva, využíva databázový systém Microsoft SQL Server 2008. Celý systém beží na vlastnom serveri firmy s operačným systémom Windows Small Business Server 2008.

Chovanie aplikácie riadiace beh informačného systému Money S5 je riadené metadátami. Zjednodušene povedané, informácie uložené do databázy obsahujú odpoveď na otázku „čo“, ale metadáta na otázku „ako“. Od základných metadát, ktoré obsahujú informácie o chovaní objektov v aplikácii (zobrazenie, spracovanie, uloženie), môže byť odvodené ľubovoľné množstvo objektov riadiacich ďalšie činnosti. Ich správa sa vykonáva pomocou internej aplikácie, ktorej súčasťou je generovanie kódu v jazyku C# a SQL skriptov pre vytváranie tabuliek a relácií v databáze. Vďaka tomuto prostrediu je systém otvorený tvorbe nových modulov certifikovanými partnermi spoločnosti, pri zachovaní jednotného vzhľadu, ovládania, stability a výkonu. S využitím metadát je tiež možné definovať celkom nové funkcie, ktoré nejde akokoľvek odlíšiť od klasických tlačových zostáv, modulov pre správu prístupových práv či fakturácií, pretože sa všetky zakladajú na rovnakom objektovom systéme.

Money S5 podporuje export a import ľubovolného zoznamu do a z dokumentu XML. Je možné nastaviť chovanie importu zvlášť pre nové záznamy (pridať/ignorovať) a zvlášť pre existujúce (prepísať/pridať/ignorovať). Pomocou automatických akcií je možné nastaviť postupnosť činností, ktoré sa uskutočňujú po pridaní alebo úprave záznamu v určitom zozname, ako napríklad export dát.

3 Možnosti komunikácie

Ako akýsi štandardizovaný formát pre výmenu dát sa presadil formát XML, ktorý sa ponúka k širokému použitiu. XML môže slúžiť ako formát pre rôzne typy dokumentov, k volaniu funkcií, alebo ako úložisko dát. Z jeho podstaty vyplýva, že vlastne môže slúžiť takmer k čomukoľvek. Formát je vďaka svojej univerzálnosti veľmi rozšírený a podporovaný. Protokol, ktorým je následne prenášaný môže byť rôzny. V dnešnej dobe sa osvedčil protokol HTTP.

3.1 XML ako základ

XML formát [9] je v súčastnosti používaný vo väčšine aplikácií a to predovšetkým ako formát pre otvorenú výmenu informácií. Ako základná myšlienka je použitie textového formátu, ktorého veľkou výhodou je zrozumiteľnosť pre človeka a počítač zároveň. Ďalšou výhodou už od jeho vzniku je používanie kódovania typu Unicode, ktoré umožňuje použitie takmer akéhokoľvek svetového jazyka.

Syntaxu XML [1] využívajú mnohé prezentačné formáty, od XHTML cez stále populárnejší vektorový grafický formát SVG, až po jazyky pre definíciu užívateľského rozhrania v moderných RIA prostrediach, ako je XAML v Silverlight a MXML vo Flash. XML dnes dominuje na poli publikovania metainformácií. Jedná sa napríklad o formáty publikovania prehľadov nových článkov, ako je RSS či Atom.

Ďalšie a pre túto prácu podstatné využitie je pre komunikáciu a predávanie dát. XML sa využíva jednak pre výmenu dát medzi backendmi jednotlivých aplikácií, ale tiež v AJAXových aplikáciach pre zasielanie aktualizácií dát do prehliadača. Moderné podnikové aplikácie potom pre samotnú komunikáciu nevyužívajú prosté XML, ale komplexnejší mechanizmus webových služieb.

XML súbor obsahuje elementy, atribúty a hodnoty, ktoré môžu ľubovolne definovať dátové typy a štruktúru dát. Každý XML dokument musí obsahovať iba jeden koreňový element, ktorému môže predchádzať hlavička, označovaná ako XML deklarácia. Hlavička určuje aká verzia XML a aké kódovanie je v dokumente použité.

3.1.1 XML schéma

Jazyk XML umožňuje používať v dokumentoch ľubovolne pomenované elementy, ale v praxi je vhodné vopred poznať aké informácie sa v dokumente môžu vyskytovať. Schéma dokumentu XML [1] umožňuje formálne definovať, aké elementy a atribúty sú v dokumente povolené. Najrozšírenejšie jazyky pre tvorbu schém sú DTD, W3C XML Schema, RELAX NG a Schematon.

Vzhľadom k tomu, že schéma jednoznačne definuje, ako môže dokument XML vypadať, môže sa použiť aj pre validáciu, čo je asi najčastejšie využitie schém. Validácia je proces, pri ktorom sa overuje, či dokument XML vyhovuje všetkým obmedzeniam, ktoré sú definované v schéme.

Niektoré jazyky pre popis schém umožňujú pre obsah jednotlivých elementov a atribútov určiť dátový typ.

3.1.2 XSL transformácia

XSL transformácia (XSLT) [1, 2] definuje programovací jazyk založený na XML, ktorý slúži k prevodu XML dokumentov na iné textové formáty. V dnešnej dobe je najobvyklejším použitím XSLT prevod XML dokumentu na iný XML dokument, čo rieši problémy nekompatibility návrhov XML dokumentov pri ich prenose medzi jednotlivými aplikáciami. Ďalším častým použitím je transformácia XML na HTML dokument alebo iný formát slúžiaci na prezentáciu dát.

Jednoduché štýly XSLT často evokujú predstavu, že XSLT je len šablónovací systém, ale XSLT je plnohodnotný programovací jazyk, ktorý ponúka bežné programátorské konštrukcie, ako premenné, podmienky, cykly a iné. XSLT ponúka tri odlišné modely programovania: model založený na vzoroch, procedurálny a deklaratívny model.

3.1.3 Objektový model XML

Ďalšou dôležitou súčasťou XML je jeho spracovanie, ktoré sa vykonáva pomocou tzv. parserov. Existujú dva základne typy: DOM (Document Object Model) a SAX (Simple API for XML). Zásadným rozdielom medzi nimi je, že SAX prechádza dokument postupne (sekvenčne) a keď narazí na používateľom definované udalosti, vykoná spätné volanie. Výhoda je predovšetkým v nízkej spotrebe operačnej pamäte a tiež v rýchlosti. Na druhú stranu je jeho použitie pomerne komplikované a preto sa oplatí len keď spracovávame veľké súbory, alebo nám záleží na výkone. Naproti tomu DOM najprv načíta celú stromovú štruktúru XML do pamäte ako objekty, ktoré reprezentujú jednotlivé časti dokumentu. S jednotlivými objektmi môžeme potom ďalej pracovať a tak zisťovať a prípadne aj upravovať obsah dokumentu. Spotreba pamäte je v tomto prípade značná a u skutočne veľkých súborov ho nie je možné ani použiť.

3.2 Webové služby založené na SOAP

Webové služby umožňujú jednoduchú komunikáciu medzi aplikáciami, napomáhajú prekonávať obmedzenia heterogénnych prostredí (napr. rôzne programovacie jazyky, v ktorých boli tieto aplikácie vyvinuté). Komunikácia je založená na platformovo nezávislých technológiách, ako je XML a protokol HTTP, ktoré robia webové služby univerzálnymi. Aplikácie si medzi sebou posielajú správy vo formáte XML, ktoré prenášajú dotazy a odpovede jednotlivých aplikácií. Klient odošle správu na server, ten ju spracuje a prípadne odošle späť odpoveď. Formát XML je v prípade webových služieb presne definovaný, rovnako sú presne definované i rozhrania cez ktoré sa k webovým službám pristupuje. Celá infraštruktúra webových služieb býva založená na troch základných technológiách:

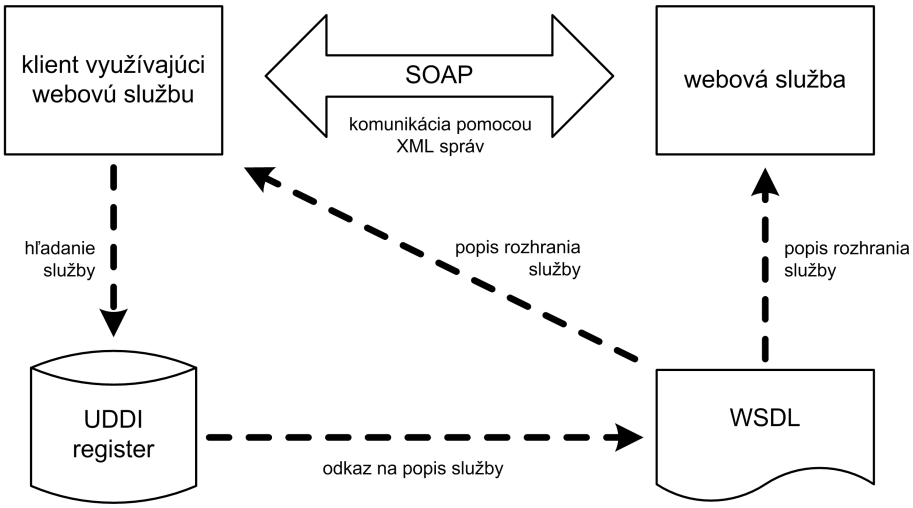
- SOAP (Simple Object Access Protocol) - jednoduchý a rozšíriteľný XML rámec na výmenu správ,
- WSDL (Web Services Description Language) - formát pre popis rozhrania webovej služby,

- UDDI (Universal Description, Discovery and Integration) - štandardný mechanizmus umožňujúci registráciu a vyhľadávanie webových služieb.

Vzájomné vzťahy [1] medzi týmito technológiami sú zachytené na obrázku 5. Protokol SOAP je jednoduchá obálka okolo samotných prenášaných dát. Avšak ponúka aj priestor pre vloženie rôznych servisných informácií, ako sú napríklad podpisy alebo prihlasovacie údaje pre prístup ku službám vyžadujúcim autentifikáciu.

Pre webovú službu väčšinou existuje popis jej rozhrania vo formáte WSDL. Súčasťou tohto popisu je aj definícia schémy vstupnej a výstupnej správy XML a informácie o tom, akým protokolom a na akej adrese je možné so službou komunikovať. Z týchto informácii už jednoznačne vyplýva, ako má vypadať soapová požiadavka.

Väčšina webových služieb si v praxi vystačí so SOAP a WSDL. Niekedy je možné sa stretnúť ešte s ďalšími nadväzujúcimi technológiami WS-*. WSDL popisy služieb je možné registrovať do UDDI registru. Ten slúži ako akýsi telefónny zoznam, ktorý umožňuje vyhľadávať služby s určitými parametrami.



Obr. 5: Princíp fungovania webových služieb založených na SOAP [1]

3.2.1 SOAP

SOAP [1, 2, 10] je rozšírením a nástupcom protokolu XML-RPC. Získal si podporu od známych softvérových spoločností, ale aj malých vývojárov softvéru. Jeho pôvodná špecifikácia 1.1 bola vydaná v roku 1999, neskôr boli niektoré funkcie rozšírené, štandard bol registrovaný konzorciom W3C a v roku 2003 bola schválená špecifikácia 1.2.

Špecifikácia definuje ako sformátovať XML, aby bolo použiteľné pre prenos vzdialených volaní a pravidla serializácie dátových typov, vrátane štruktúrovaných dátových typov a reťazcov. Taktiež popisuje ako používať transportné protokoly aplikačnej vrstvy k prenosu správ, ako sú napríklad HTTP, SMTP alebo FTP. V súčasnosti poskytuje priame nadviazanie na HTTP, pretože ide o všeobecne zaužívaný štandard a vďaka tomu môže

SOAP jednoducho prechádzať cez firewally, čo nie je u iných protokolov samozrejmosťou. XML formát bol zvolený hlavne kvôli širokej podpore, rozšíriteľnosti a existencii veľkého množstva voľne dostupných nástrojov pre prácu s ním, čo zjednodušuje nasadenie SOAP. Častou námietkou kritikov býva zdĺhavá syntax XML, čo spôsobuje vyššie nároky na prenos i strojove spracovanie. Výhodou je čitateľnosť pre človeka. Riešením tohto problému môže byť napríklad vytvorenie binárnej formy XML.

SOAP správy sa skladajú z elementu `Envelope` (obálka okolo celej správy) s voliteľným detským elementom `Header` (hlavička správy pre prenos servisných informácií) a povinným detským elementom `Body` (telo správy pre prenos samotného obsahu správy). Tieto elementy patria do špeciálneho menného priestoru `http://www.w3.org/2003/05/soap-envelope`. Telo správy môže obsahovať ľubovoľnú XML syntax, ale syntax celej správy je presne definovaná pomocou XML schémy. Výpis 1 je ukážkou jednoduchej požiadavky zasielanej webovej službe.

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap−envelope">
  <soap:Header>
    ...
  </soap:Header>
  <soap:Body>
    <Numbers>
      <aNum>10</aNum>
      <bNum>20.5</bNum>
    </Numbers>
  </soap:Body>
</soap:Envelope>
```

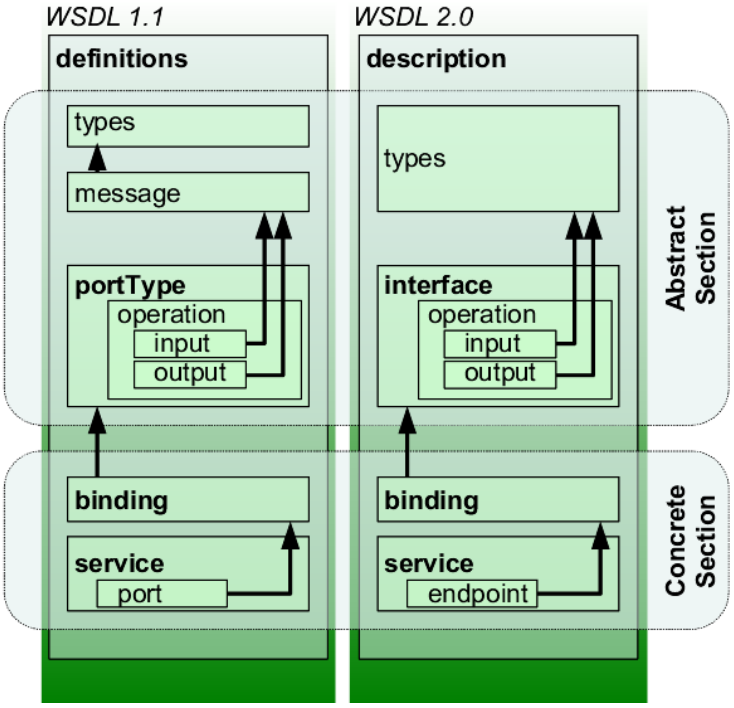
Výpis 1: Ukážka jednoduchej požiadavky SOAP

V súčasnosti existujú dva základné štýly [12, 13] na výmenu správ SOAP – *Document* a *RPC*. Tzv. dokumentový štýl znamená, že telo správy jednoducho obsahuje dokument XML, z ktorého formátom musí súhlasiť odosielateľ i prijímateľ. Na druhej strane štýl RPC znamená, že telo správy obsahuje reprezentáciu XML metódy volania. Existujú však dve techniky pre rozhodnutie, ako serializovať dáta v rámci tela: použitie základných definícií XML schémy (*literal*) a použitie enkódovacích pravidiel SOAP (*encoded*). S prvým prístupom určuje definícia schémy XML formát pre telo správy bez nejasností. S druhým prístupom musí SOAP spracovateľ prejsť počas spracovania rôznymi enkódovacími pravidlami SOAP, aby zistil správnu serializáciu v rámci tela. Táto technika je viac náchylná na chyby a problémy s interoperabilitou. Oveľa bežnejšie je používať dokumentový štýl so základnými definíciami schémy (*Document/literal*) a štýl RPC so SOAP enkódovacími pravidlami (*RPC/encoded*). *Document/encoded* a *RPC/literal* sú prístupné, no nie sú bežné a nedávajú ani zmysel. *Document/literal* je štýl, na ktorý sa zameriava väčšina platforiem pre webové služby.

3.2.2 WSDL

Jazyk WSDL [1, 11] slúži k popisu webových služieb ako množiny koncových bodov spracovávajúcich správy. Samotné správy a operácie s nimi sú pritom popisované na abstraktnej úrovni a až neskôr sú zviazané s konkrétnym sieťovým protokolom a dátovým formátom. To umožňuje ľahké vytvorenie popisu rozhrania, ktoré ponúka jednu službu niekoľkými rôznymi spôsobmi. Samozrejme v praxi sa pomocou WSDL najčastejšie popisujú služby, ktoré posielajú správy pomocou formátu SOAP a protokolu HTTP.

WSDL súbor s definíciou rozhrania služby je dokument XML. Na obrázku 6 je znázornená jeho štruktúra. WSDL vzniklo ako spoločná iniciatíva firiem Microsoft a IBM, ktoré si uvedomili potrebu zjednotenia jazyka používaného pre popis rozhrania webových služieb. Súčasná verzia špecifikácie je 2.0 (premenovaná 1.2) a je spravovaná konzorciom W3C. Oproti verzii 1.1 umožňuje zviazanie aj s ostatnými metódami HTTP požiadaviek, ako napríklad PUT či DELETE a preto ponúka lepšiu podporu pre tzv. RESTful webové služby a tým ich ľahšiu implementáciu. Napriek tomu je podpora pre túto špecifikáciu so strany softvérových nástrojov k webovým službám stále nedostatočná.



Obr. 6: Štruktúra WSDL 1.1 a WSDL 2.0 [11]

3.3 Webové služby založené na REST

Webové služby založené na SOAPu bývajú často kritizované za to, že veci zbytočne komplikujú. Namiesto zasielania jednoduchých dokumentov XML sa všetko obaľuje obálkou SOAP, nehovoriac o komplikovanom používaní ďalších WS-* technológií. Pre jednoduchšie scenáre sa preto do obľuby dostávajú tzv. REST webové služby (Representational State Transfer). Jednotlivé operácie webovej služby sú mapované na metódy protokolu HTTP, ako je GET, POST, PUL a DELETE. Data požiadavky sú potom prenášané ako dokumenty XML. V jednoduchších prípadoch môžu byť parametre požiadavky prenášané ako parametre metódy GET v URL adrese požiadavky. [1]

Zatiaľ čo SOAP odpovedajú princípu RPC, teda volaniu procedúr, REST služby sú orientované na zdroje - dáta. Tento rozdiel sa prejavuje aj v URL, v prípade SOAP bude obsahovať typicky sloveso (*vratObjednavku*), ale v prípade REST služby bude obsahovať len podstatné meno (*objednavka*). [15]

Rozhranie REST je použiteľné pre jednotný a ľahký prístup ku zdrojom (resources). Všetky zdroje majú vlastný identifikátor URI a REST definuje štyri základné metódy pre prístup k nim, ktoré sú známe pod označením CRUD. Teda vytvorenie dát (create), získanie požadovaných dát (retrieve), zmenu (update) a zmazanie (delete). Tieto metódy sú implementované pomocou odpovedajúcich metód HTTP protokolu: GET (retrieve), POST (create), DELETE, PUT (update). [14]

3.4 AJAX

AJAX (Asynchronous JavaScript and XML) [1] je dnes veľmi populárna technika pri vývoji veľmi interaktívnych webových aplikácií. Základný princíp AJAXu je pritom veľmi jednoduchý - dovoľuje aktualizovať iba časť stránky zobrazenej vo webovom prehliadači bez nutnosti načítania a vykresľovania celej stránky. Na rozdiel od klasických webových aplikácii poskytujú užívateľský príjemnejšie prostredie, ale vyžadujú použitie moderných webových prehliadačov.

AJAX dovoľuje poslať pomocou JavaScriptu oddelenú HTTP požiadavku a dáta, ktoré nám server vráti, potom môžeme v JavaScripte ľubovoľne ďalej spracovávať, napríklad ich zakomponovať do práve zobrazenej stránky.

Základom AJAXu je možnosť čítať pomocou JavaScriptu dáta z webového servera. K tomu slúži objekt *XMLHTTPRequest*, ktorý dovoľuje zaslať HTTP požiadavku a prečítať následne výsledok.

Server dáta AJAXovej aplikácii posiela vo formáte XML, ktorý je možné na klientovi spracovať pomocou rozhrania DOM. Mnoho aplikácií ale používa jednoduchší formát JSON (JavaScript Object Notation), alebo zasiela priamo spustiteľný kód v JavaScripte.

Pri vývoji rozsiahlejších AJAXových aplikácií býva bežné využívať hotové JavaScriptové knižnice, ktoré odtienia implementačné rozdiely jednotlivých prehliadačov a uľahčia niektoré operácie - napríklad vylepšia niekedy ťažkopádne metódy rozhrania DOM. Takých knižníc v dnešnej dobe existujú desiatky, v tomto projekte je využitá knižnica jQuery, obľúbená pre svoju komplexnosť.

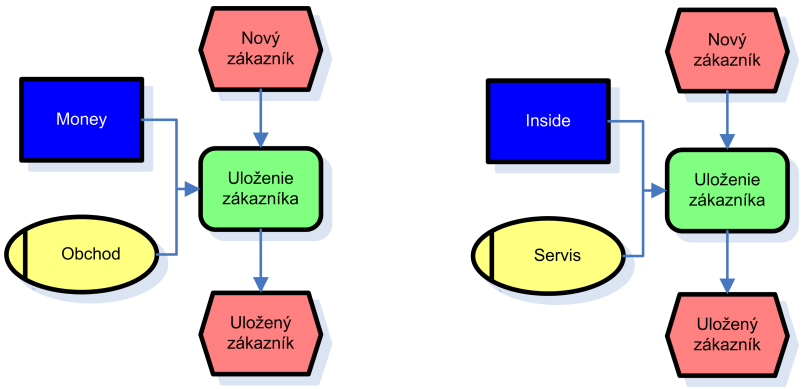
4 Špecifikácia zadania

Cieľom práce je navrhnúť a realizovať rozšírenie informačného systému Inside. V rámci rozšírenia je najdôležitejšie jeho prepojenie s ERP systémom Money S5 a webovou stránkou firmy. Používaním dvoch nezávislých systémov vo firme sa časom prišlo na to, že mnohé spravované údaje a činnosti sa zbytočne niekoľko krát opakujú. Tieto procesy bolo potrebné špecifikovať, opísať a následne navrhnúť a realizovať ich efektívnejšie riešenie. Pri návrhu treba zohľadniť, že firma plánuje vytvorenú aplikáciu Inside ponúkať ďalším firmám, ktoré používajú v rámci firmy odlišné systémy a preto riešenie prepojenia a komunikácie by malo byť čo najuniverzálnejšie.

4.1 Špecifikácia biznis modelov

K špecifikácii vybraných biznis procesov použijeme neformálny prístup textového popisu a následne formálny prístup pomocou vývojového diagramu EPC (Event-driven Process Chain).

Na obrázku 7 je veľmi jednoduchý EPC diagram, ktorý znázorňuje uloženie nového zákazníka. Uloženie prebieha v oboch systémoch nezávisle na sebe. Technik v servisnom oddelení firmy ukladá zákazníka do systému Inside a pracovník obchodného oddelenia ukladá zákazníka do systému Money. Už pri ukladaní sa teda v jednom z oddelení zbytočne plytvá čas, ako na strane pracovníka, tak na strane zákazníka. Uložené údaje sú na sebe úplne nezávisle a časom dochádza aj k ich nekonzistencii. Následné vyhľadávanie zákazníka prebieha len v dátach použitého systému.

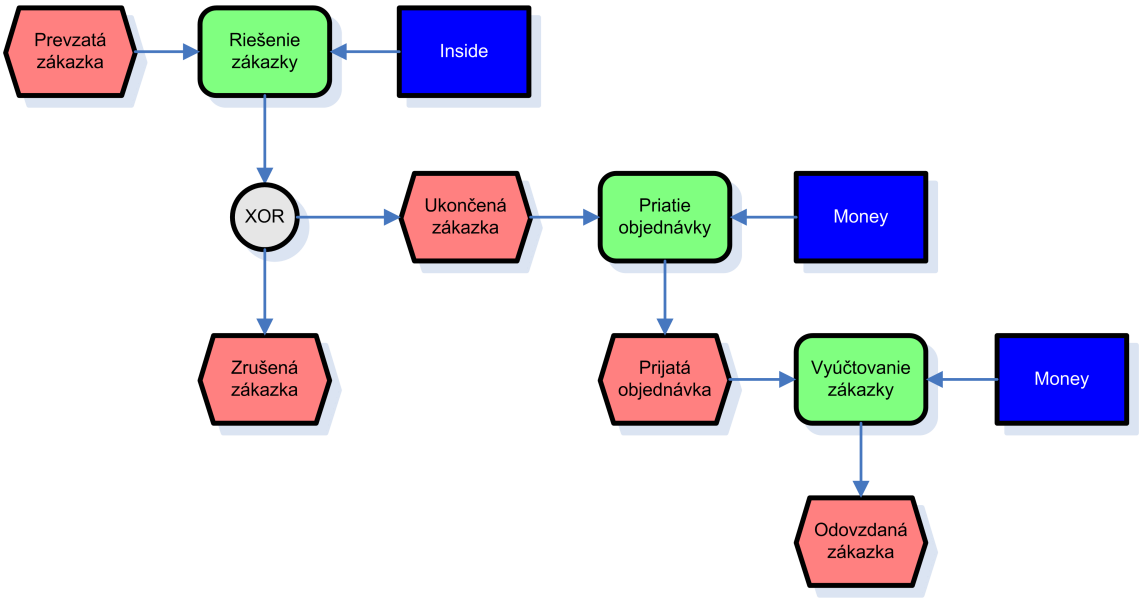


Obr. 7: EPC - Uloženie nového zákazníka

Požiadavkou zadávateľa teda je, aby údaje o zákazníkovi boli vo firme ukladané iba jeden krát. Ak ich uloží obchodné oddelenie do systému Money, informácie budú prístupné aj v systéme Inside a naopak. Samozrejme sa musí počítať aj s prípadnou úpravou údajov, či vymazaním zákazníka.

Na obrázku 8 je ďalší EPC diagram, znázorňujúci riešenie a následné vyúčtovanie servisnej zákazky. Po prevzatí servisným strediskom je zákazka riešená a všetky informácie sú zaznamenávane do systému Inside. V priebehu riešenia môže byť zákazka z rôznych

dôvodov zrušená a teda vrátená zákazníkovi. Ak ale bola úspešne ukončená (opravená) je potrebné jej vyúčtovanie (zaplatenie), ktoré prebieha v obchodnom oddelení a systéme Money. Pri ukončení zákazky technik vytlačí so systému Inside pracovný list so zoznamom použitého materiálu a vykonanej práce. Podľa neho potom môže obchodník prijať objednávku a na jej základe prebieha vyúčtovanie, či už platbou v hotovosti alebo fakturáciou. Informácie o použitom materiále a vykonanej práci sú teda v rámci firmy ukladané minimálne dva krát. To je pomerne nepraktické, zdĺhavé a môže tiež dochádzať k chybám pri prepise. Ešte vážnejším problémom ale je, že ak technik použije nejaký materiál v priebehu riešenia zákazky, údaje uloží do systému Inside, ale obchodník nemá žiaden prehľad o použitom materiále a naďalej mu na sklade v systéme Money zobrazuje materiál k dispozícii.

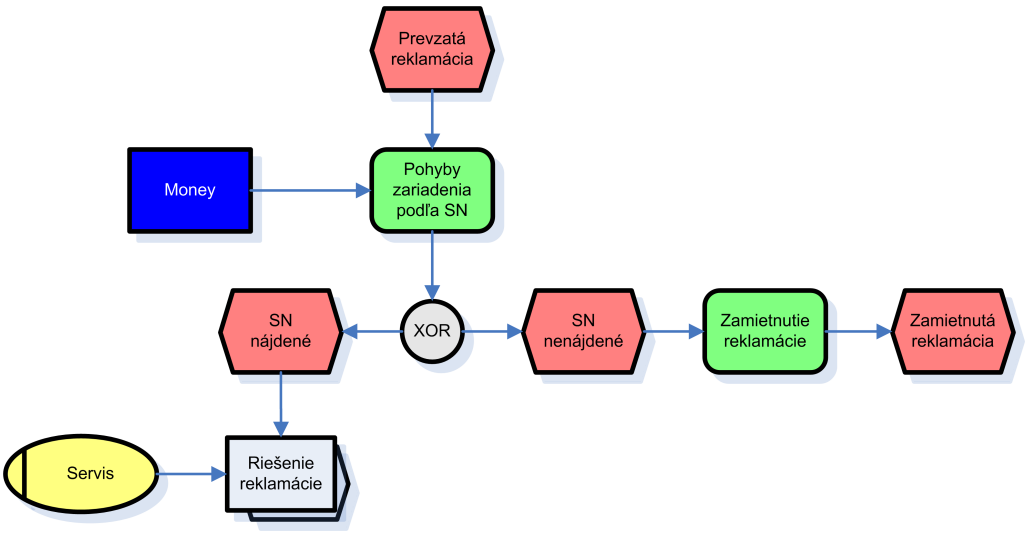


Obr. 8: EPC - Riešenie a vyúčtovanie servisnej zákazky

Požiadavkou teda je, aby servisná zákazka v systéme Inside, bola automatický evidovaná v systéme Money ako prijatá objednávka. V priebehu riešenia zákazky a využitia materiálu, sa všetko zaznamená aj v objednávke a tým pádom by bol materiál v systéme Money označený na sklade ako rezervovaný zákazkou. Po ukončení zákazky by mal obchodník pristúpiť priamo k vyúčtovaniu zákazky.

Na obrázku 9 je znázornený EPC diagram vyhľadania skladových pohybov zásoby podľa výrobného čísla pri reklamáciách. Tovar vo firme býva jednoznačne identifikovaný podľa svojho výrobného čísla (tzv. SN). Vrámci skladového hospodárstva vedeného v systéme Money, sú evidované skladové pohyby zásoby (tovaru), ako sú kúpa, príjem, predaj, výdaj a iné. Ak je servisným oddelením prevzatá zákazka typu reklamácia, je potrebné vyhľadať tieto pohyby podľa výrobného čísla reklamovaného zariadenia. Jednak v prípade ak zákazník nevie predložiť doklady o kúpe a záruke a taktiež v prípade, keď technik potrebuje zistiť u ktorého dodávateľa a kedy bolo zariadenie zakúpené, aby

vedel správne ďalej postupovať. V prípade nenájdenia čísla v systéme, je reklamácia zamietnutá. V opačnom prípade je ďalej riešená servisným oddelením. Keďže sa tieto informácie evidujú v systéme Money, musí technik o overenie požiadať pracovníka obchodného oddelenia. Takáto sprostredkovaná aktivita je veľmi nepraktická.



Obr. 9: EPC - Vyhľadanie skladových pohybov zásoby podľa SN

Požiadavkou zadávateľa teda je, aby servisný technik mohol vyhľadať skladové pohyby zariadenia podľa výrobné čísla priamo so systému Inside a tým úplne vynechať pomoc od obchodného oddelenia.

V úvode kapitoly spomenuté prepojenie systému Inside na webovú stránku firmy by sa malo stať ďalšou službou pre zákazníkov. Zákazník by si mohol cez webovú stránku kedykoľvek pozrieť aktuálny stav svojej zákazky, kontakt na zodpovedného technika, informácie o vykonanej práci a ďalšie zaujímave údaje. Momentálne o tieto informácie musí požiadať osobne, telefonicky, či emailom a pracovník firmy musí odpoveď pohľadať v systéme.

Aj po konzultáciách s vývojármi spoločnosti Cígler Software, sme došli k záveru, že najlepšie riešenie pre presun údajov do systému Money je ich import pomocou dokumentov XML. Importom nie je nijako obchádzaná aplikačná logika systému a údaje môžu byť systémom kontrolované a spracovávané. Tento postup vyhovuje aj podmienke univerzálnosti riešenia, pretože väčšina moderných informačných systémov umožňuje prácu s XML. Procesy importu a exportu dát ale bude potrebné zautomatizovať, aby bežali na pozadí systémoch a nebola potrebná akákoľvek interakcia od používateľa.

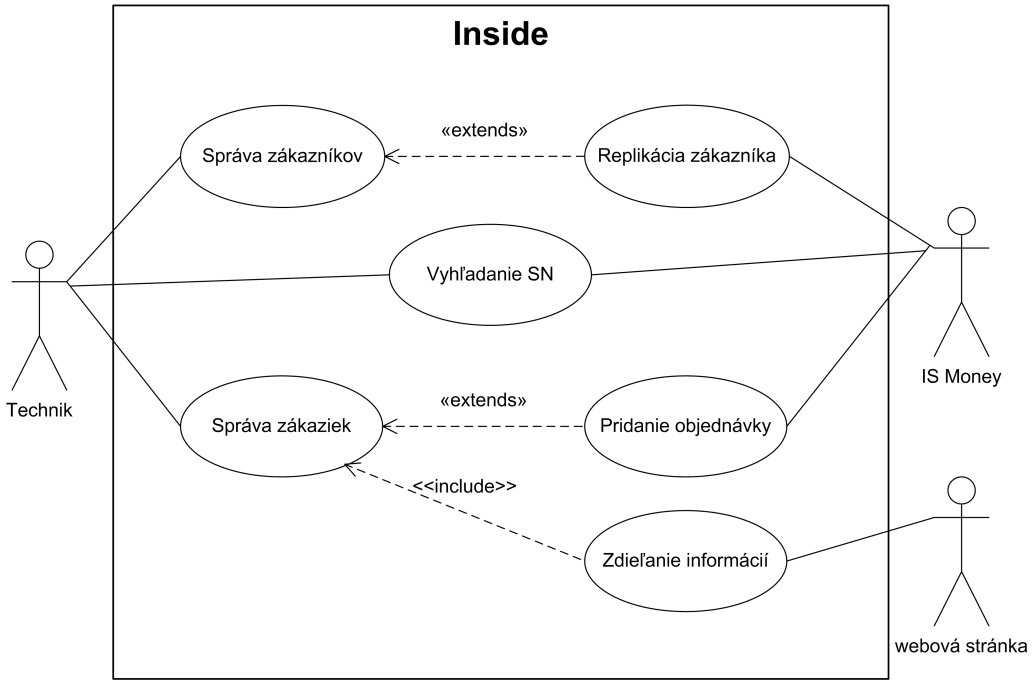
Pri vyhľadávaní pohybov zásob bude ale prínosnejšie vytvorenie a použitie webovej služby komunikujúcej priamo s databázou, aj napriek tomu, že toto riešenie bude bez výrazných úprav fungovať len s dátami z Money. Akákoľvek replikácia týchto objemných dát by prácou prevýšila prínos.

5 Špecifikácia požiadaviek

Cieľom špecifikácie požiadaviek je prostredníctvom funkcionality systému popísať čo má softvérový systém vykonávať. Modely špecifikácie požiadaviek slúžia k odsúhlaseniu zadania medzi vývojovým tímom a zadávateľom. [4]

5.1 Funkčná špecifikácia

Funkčnú špecifikáciu budeme definovať prostredníctvom diagramu Use Case. Diagramy Use Case [5] definujú vzťahy medzi prípadmi užitia systému a aktérmi v okolí systému. Prípady užitia špecifikujú vzory chovania realizované softvérovým systémom. Každý prípad užitia môžeme chápať ako postupnosť vzájomne nadväzujúcich transakcii vykonaných v dialógu medzi aktérom a systémom. Ako aktérov potom definujeme užívateľov či iné systémy, ktoré budú vstupovať do interakcie s vyvíjaným systémom.



Obr. 10: Use Case - Systém Inside

5.1.1 Systém Inside

Use Case diagram na obrázku 10 popisuje rozšírenie systému Inside o novú funkcionalitu. Správa zákazníkov bude môcť využiť tzv. replikáciu zákazníka. Pri pridávaní, úprave, či vymazaní zákazníka v Inside, budú informácie prenesené aj do systému Money. Rovnako tak pri zmene v systéme Money budú informácie prenesené do systému Inside. Správa zákaziek bude môcť využiť pridanie objednávky. Pri spravovaní alebo

ukončení zákazky budú potrebné informácie uložené ako prijatá objednávka v systéme Money. Technik bude mať k dispozícii v systéme Inside možnosť vyhľadania skladových pohybov zásoby podľa výrobného čísla, ktoré sa evidujú v systéme Money. Webová stránka firmy bude môcť využiť zdieľanie informácií o aktuálnom stave a riešení zákazky.

5.1.1.1 Replikácia zákazníka

Primárny aktér: Technik
Vstupné podmienky:

- pridanie zákazníka
- úprava zákazníka
- vymazanie zákazníka

Záruka úspechu:

- zhodné informácie o zákazníkovi v oboch systémoch

Hlavný scenár:

1. Technik uloží údaje o zákazníkovi.
2. Inside vyexportuje informácie o zákazníkovi do Money.
3. Money importuje informácie o zákazníkovi z Inside.
4. Money uloží údaje o zákazníkovi.
5. Money vyexportuje Money ID zákazníka do Inside.
6. Inside importuje informácie o zákazníkovi z Money.
7. Inside uloží údaje o zákazníkovi.

Rozšírenia:

5. Údaje sú iba upravované alebo vymazávané.
   (a) Koniec operácie.

5.1.1.2 Pridanie objednávky

Primárny aktér: Technik
Vstupné podmienky:

- riešenie zákazky
- ukončenie zákazky

Záruka úspechu:

- uložená prijatá objednávka s použitým materiálom v systéme Money

Hlavný scenár:

1. Technik uloží údaje o zákazke.
2. Inside vyexportuje informácie o objednávke do Money.
3. Money importuje informácie o objednávke z Inside.
4. Money uloží objednávku.

Rozšírenia:

4. Objednávka už existuje.
   (a) Money vymaže pôvodnú objednávku.
   (b) Money uloží novú objednávku.

5.1.1.3 Vyhľadanie výrobného čísla

Primárny aktér: Technik
Vstupné podmienky:

- prevzatá reklamácia

Záruka úspechu:

- zobrazené informácie o skladových pohyboch zariadenia

Hlavný scenár:

1. Technik dá vyhľadať zariadenie podľa výrobného čísla.
2. Inside vytvorí požiadavku na systém Money.
3. Money spracuje požiadavku.
4. Money vyhľadá zariadenie.
5. Money vytvorí odpoveď pre systém Inside.
6. Inside spracuje odpoveď.
7. Inside zobrazí informácie.

Rozšírenia:

4. Zariadenie nenájdené.
   (a) Money vráti v odpovedi chybu.

5.1.1.4 Zdieľanie informácií

Primárny aktér: Webová stránka
Vstupné podmienky:

- zákazník vyhľadáva informácie o svojej zákazke

Záruka úspechu:

- zobrazené informácie o aktuálnom stave a riešení zákazky

Hlavný scenár:

1. Zákazník dá vyhľadať zákazku podľa čísla.
2. Webová stránka vytvorí požiadavku na systém Inside.
3. Inside spracuje požiadavku.
4. Inside vyhľadá zákazku.
5. Inside vytvorí odpoveď pre webovú stránku.
6. Webová stránka spracuje odpoveď.
7. Webová stránka zobrazí informácie.

Rozšírenia:

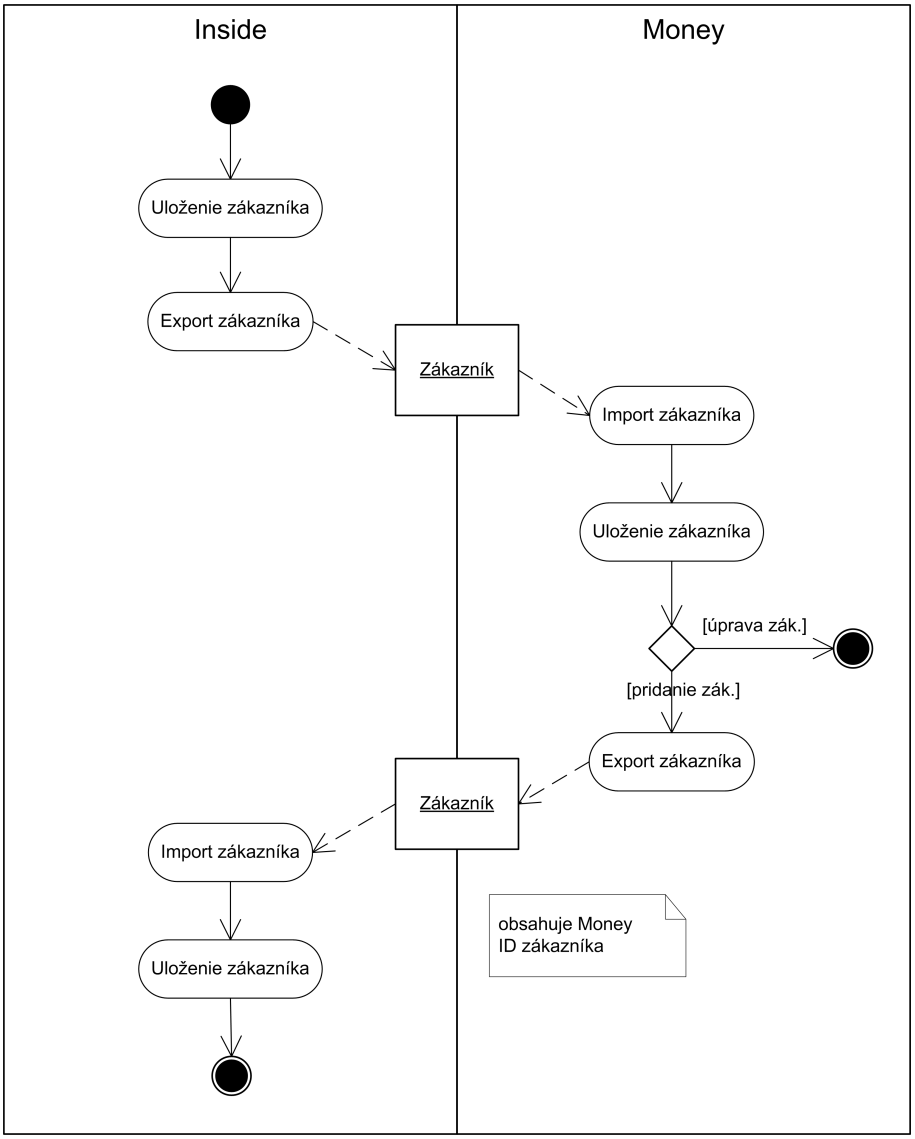
4. Zákazka nenájdená.
   (a) Inside vráti v odpovedi chybu.

5.2 Špecifikácia chovania

Podstata špecifikácie chovania [5] objektovo orientovaného systému spočíva v modelovaní komunikácie medzi objektmi. Práve tieto interakcie vedú v konečnom dôsledku k požadovanému chovaniu celého systému - k zaisteniu jeho požadovanej funkcionality. Diagram aktivít popisuje toky činnosti pomocou aktivít reprezentujúcich stavy a prechodmi medzi nimi. Ďalším účelom diagramu aktivít je definovať, ktorý objekt zodpovedá za danú aktivitu, prípadne aké objekty sú aktivitami vytvárané, spotrebované alebo modifikované. Týmto spôsobom sú do jedného diagramu premietnuté nielen toky riadenia, ale tiež dátové toky.

5.2.1 Replikácia zákazníka

Replikácia zákazníkových údajov začína ich uložením, či už pri pridávaní nového zákazníka, alebo úprave jeho stávajúcich údajov. Úpravou môžme chápať aj vymazanie. Po uložení údajov, systém Inside exportuje informácie do systému Money. Ten údaje importuje a zákazníka uloží do svojej databázy. Ak sa jednalo o úpravu zákazníkových údajov, proces je týmto ukončený. Ak sa ale jedná o pridanie nového zákazníka, je potrebné informovať systém Inside o pridelenom jednoznačnom identifikátore (ID) v systéme Money. To znamená, že systém Money exportuje informácie späť do systému Inside. Ten údaje importuje a zmeny zákazníka (Money ID) uloží. Opísaný proces je znázornený diagramom aktivít na obrázku 11.

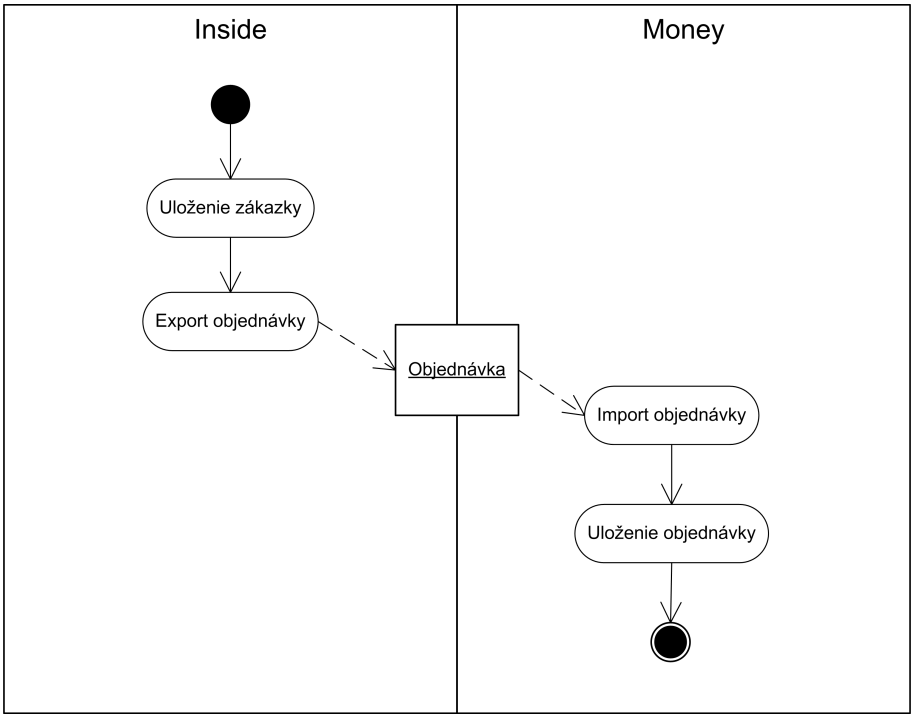


Obr. 11: Diagram aktivít - Replikácia zákazníka

Rovnako, ale v prehodených úlohách bude prebiehať aj uloženie údajov zákazníka zo systému Money do systému Inside.

5.2.2 Pridanie objednávky

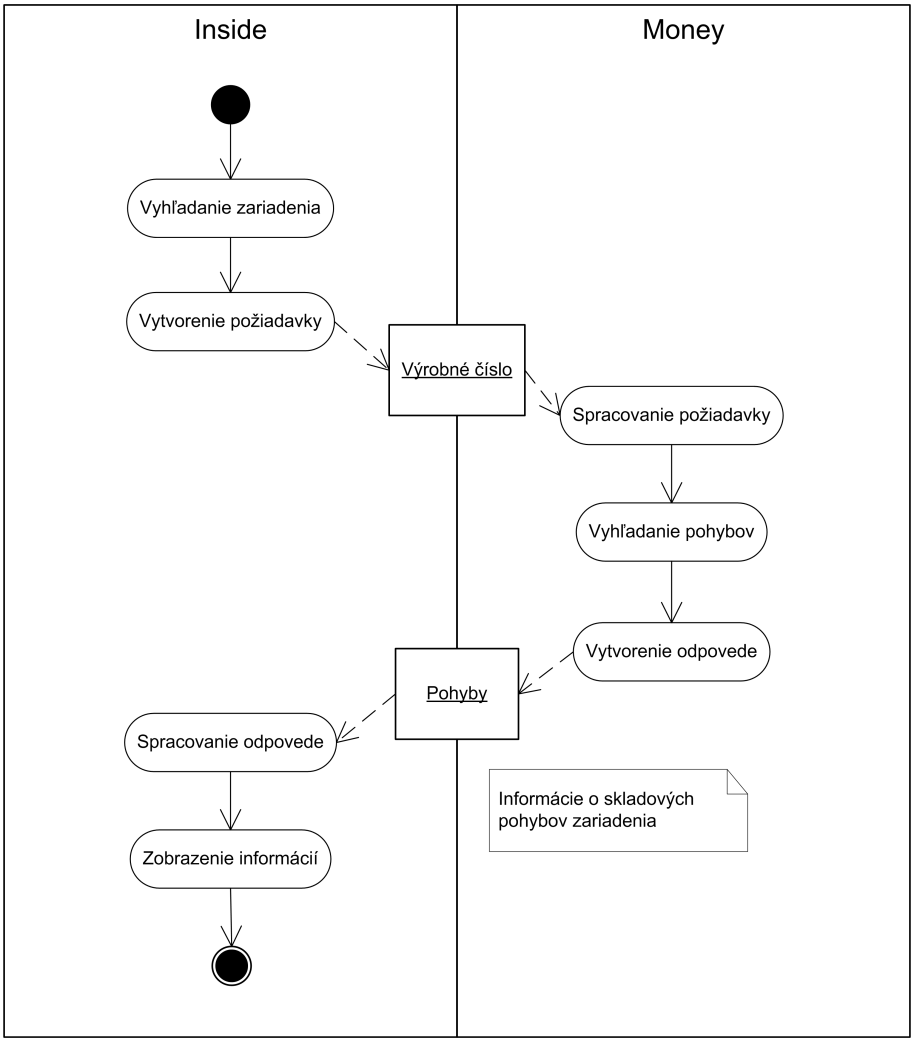
Pridanie objednávky začína uložením zákazky v systéme Inside, pri jej riešení alebo ukončení. Po uložení údajov systém Inside exportuje informácie do systému Money. Ten údaje importuje a objednávku uloží. Opísaný proces je znázornený diagramom aktivít na obrázku 12.



Obr. 12: Diagram aktivít - Pridanie objednávky

5.2.3 Vyhľadanie výrobného čísla

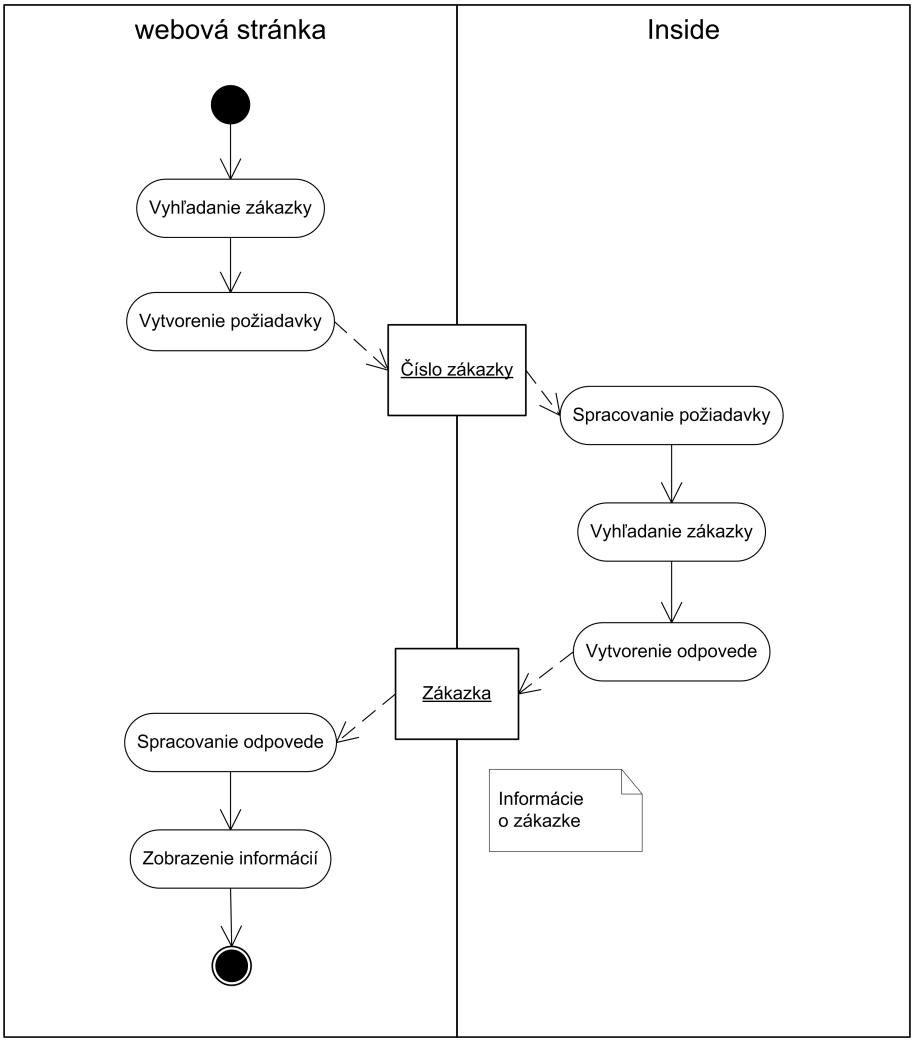
Vyhľadanie informácií o skladových pohyboch zariadenia podľa výrobného čísla začne požiadavkou od technika na jeho vyhľadanie. Systém Inside následne vytvorí požiadavku, ktorej obsahom je výrobné číslo a tú odošle systému Money. Ten požiadavku spracuje a podľa výrobného čísla vyhľadá informácie o zariadení. Podľa výsledku vyhľadania vytvorí odpoveď, ktorú zašle späť. Systém Inside odpoveď spracuje a zobrazí používateľovi. Opísaný proces je znázornený diagramom aktivít na obrázku 13.



Obr. 13: Diagram aktivít - Vyhľadanie výrobného čísla

5.2.4 Zdieľanie informácií

Zobrazenie informácií o aktuálnom stave a riešení zákazky začína vyhľadaním zákazky podľa jej čísla od zákazníka na webovej stránke firmy. Webová stránka následne vytvorí požiadavku, ktorej obsahom je číslo zákazky a tú odošle systému Inside. Ten požiadavku spracuje a podľa čísla zákazky vyhľadá informácie o zákazke. Podľa výsledku vyhľadania vytvorí odpoveď, ktorú zašle späť. Webová stránka odpoveď spracuje a zobrazí zákazníkovi. Opísaný proces je znázornený diagramom aktivít na obrázku 14.



Obr. 14: Diagram aktivít - Zdieľanie informácií

6 Analýza

Po špecifikácii funkcionality rozšírení systému je nutné určiť, ako táto funkcionalita bude riešená v implementačnej fáze. Vytvorením modelu analýzy môžme správne pochopiť štruktúry a funkcie systému. Model analýzy skúma špecifikované požiadavky z pohľadu objektov, ktoré je možné nájsť v riešenej doméne. Tieto objekty zodpovedajú za plnenie požadovanej funkcionality poskytovaním svojich služieb v komunikácii s inými objektmi. [4]

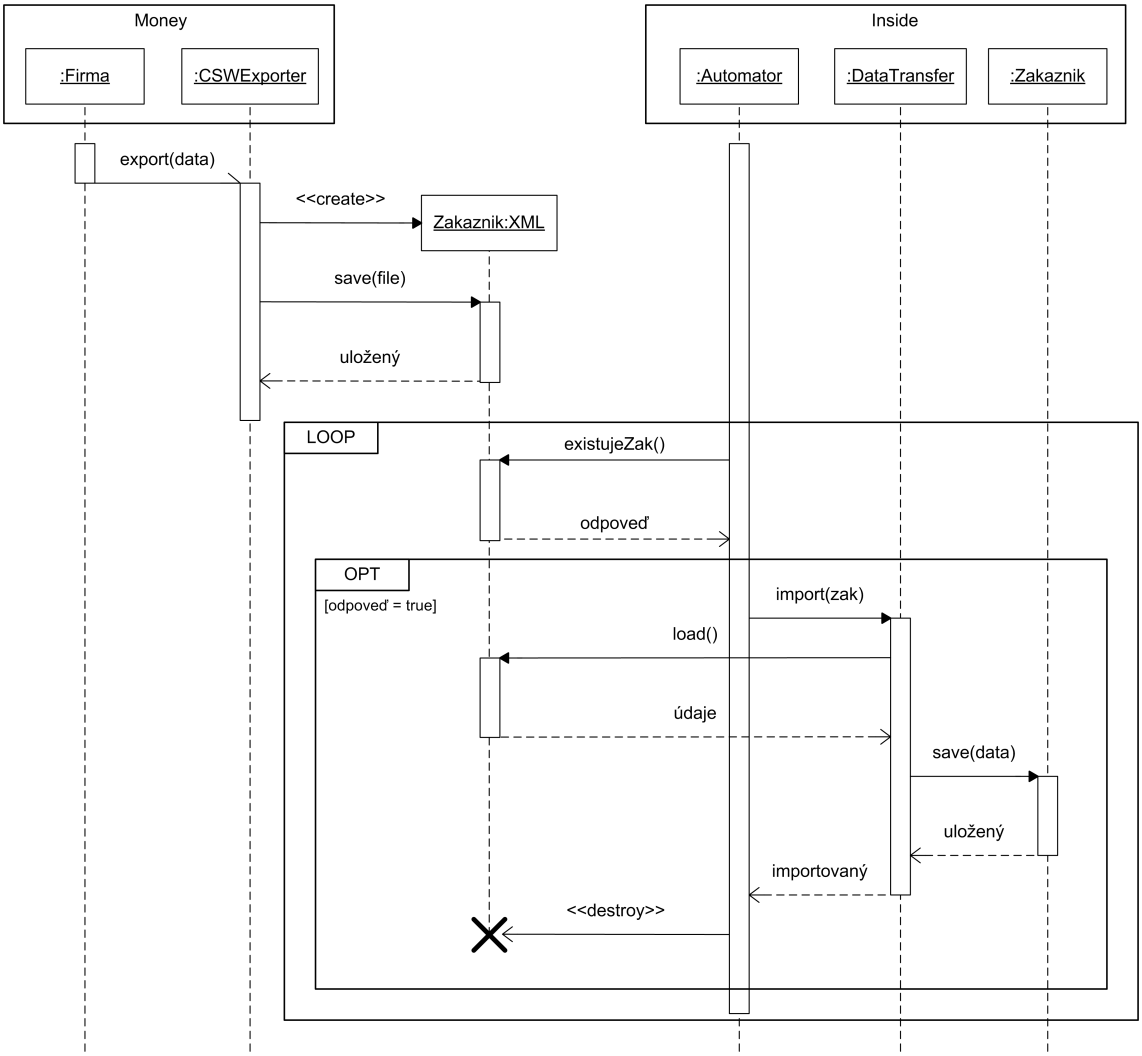
6.1 Model objektovej spolupráce

Vo vzťahu k definovaným prípadom užitia je nutné definovať vzájomnú spoluprácu medzi objektmi, ktorá povedie k splneniu ich funkcionality, účelu ku ktorému boli navrhnuté. Pre modelovanie tejto spolupráce použijeme sekvenčné diagramy, ktoré poskytujú možnosti k tomu, aby znázornili, ako medzi sebou objekty spolupracujú. Sekvenčný diagram nie je možné nakresliť pre všetky scenáre prípadov užitia. Prácnosť s tým spojená by prevýšila prínos. Diagramy preto použijeme len pre zaujímave a prácu ozrejmujúce scenáre.

6.1.1 Replikácia zákazníka

Na obrázku 15 je znázornený sekvenčný diagram, ktorý zobrazuje jeden z možných scenárov prípadu užitia Replikácia zákazníka. Konkrétne sa jedná o export údajov zákazníka po ich zmene v systéme Money a ich import a uloženie v systéme Inside. Tento scenár je realizovaný vzájomnou interakciou medzi objektami v systéme Money a medzi objektami v systéme Inside. Vzájomnú interakciu začína objekt firmy v systéme Money správou *export()* na objekt CSWExporter pre exportovanie zmenených zákazníkových údajov. Objekt CSWExporter vytvára objekt XML označený ako zákazník a požiadavkou *save()* do neho ukladá dáta. V systéme Inside je aktívny objekt Automator, ktorý v cykle v pravidelných intervaloch kontroluje, či existuje objekt XML. Keď zistí, že objekt existuje požiadavkou *import()* na objekt DataTransfer, zaistí importovanie objektu. Objekt DataTransfer požiadavkou *load()* načíta objekt XML a následne získané údaje požiadavkou *save()* na objekt Zákazník uloží. Po úspešnom importe môže DataTransfer vymazať objekt XML.

Pri ukladaní nového zákazníka, by po skončení scenára, nasledovala jeho kópia, ale so systémami v opačných úlohách. V opačných úlohách sú systémy tiež pri ukladaní zákazníka zo systému Inside do systému Money. Veľmi podobný scenár predstavuje sekvenčný diagram Pridanie objednávky.

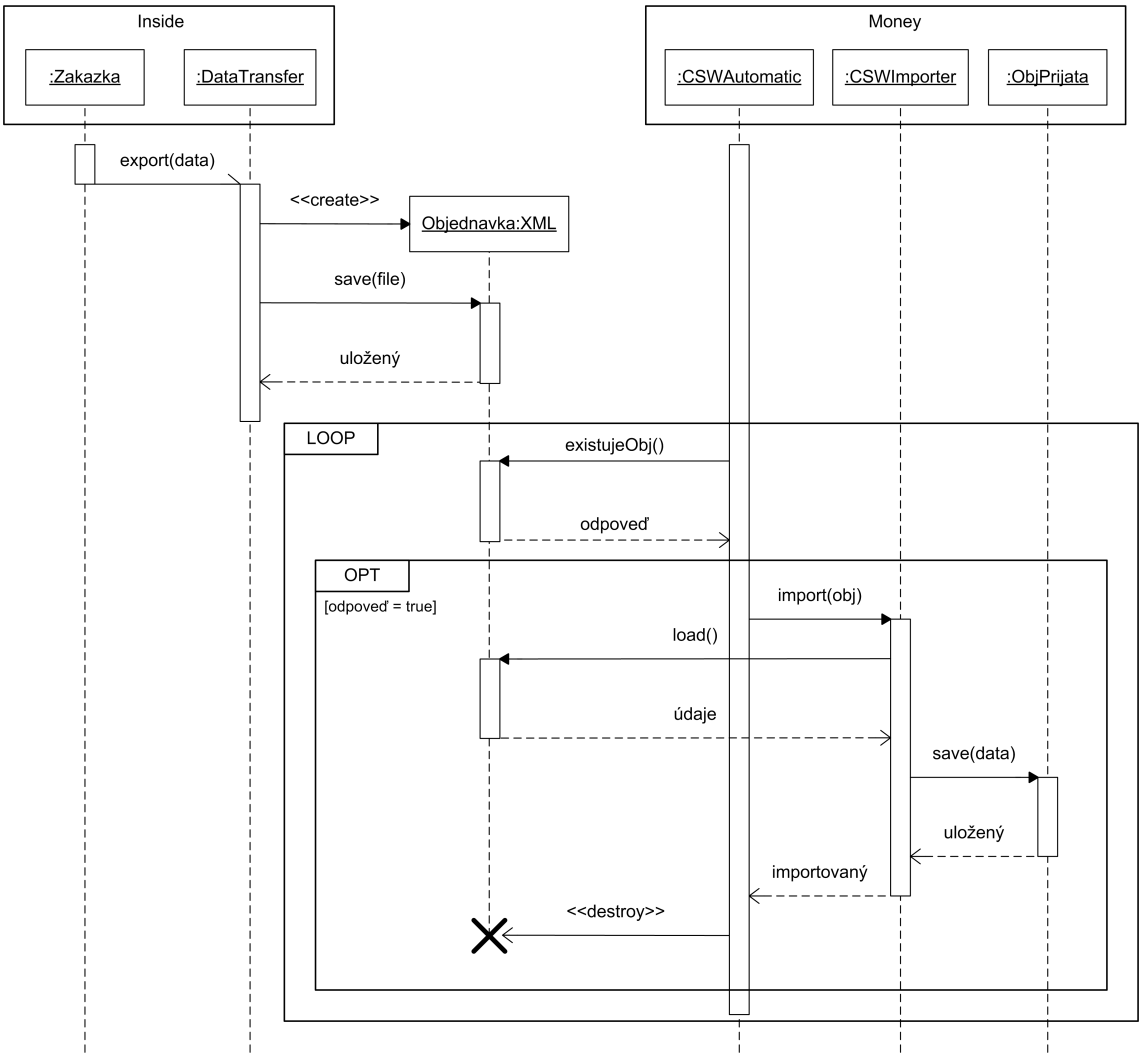


Obr. 15: Sekvenčný diagram - Replikácia zákazníka

6.1.2 Pridanie objednávky

Na obrázku 16 je znázornený sekvenčný diagram, ktorý zobrazuje scenár prípadu užitia Pridanie objednávky. Jedná sa o export údajov o použitom materiále pri zákazke, keď sú tieto informácie uložené do systému Inside. Následné sú importované a uložené v systéme Money. Vzájomnú interakciu začína objekt zákazky v systéme Inside správou *export()* na objekt DataTransfer pre exportovanie údajov zo zákazky. Objekt DataTransfer vytvára objekt XML označený ako objednávka a požiadavkou *save()* do neho ukladá dáta. V systéme Money je aktívny objekt CSWAutomatic, ktorý v cykle v pravidelných intervaloch kontroluje, či existuje objekt XML. Keď zistí, že objekt existuje požiadavkou *import()* na objekt CSWImporter, zaistí importovanie objektu. Objekt CSWImporter

požiadavkou *load()* načíta objekt XML a následne získané údaje požiadavkou *save()* na objekt ObjednavkaPrijata uloží. Po úspešnom importe môže CSWAutomatic vymazať objekt XML.

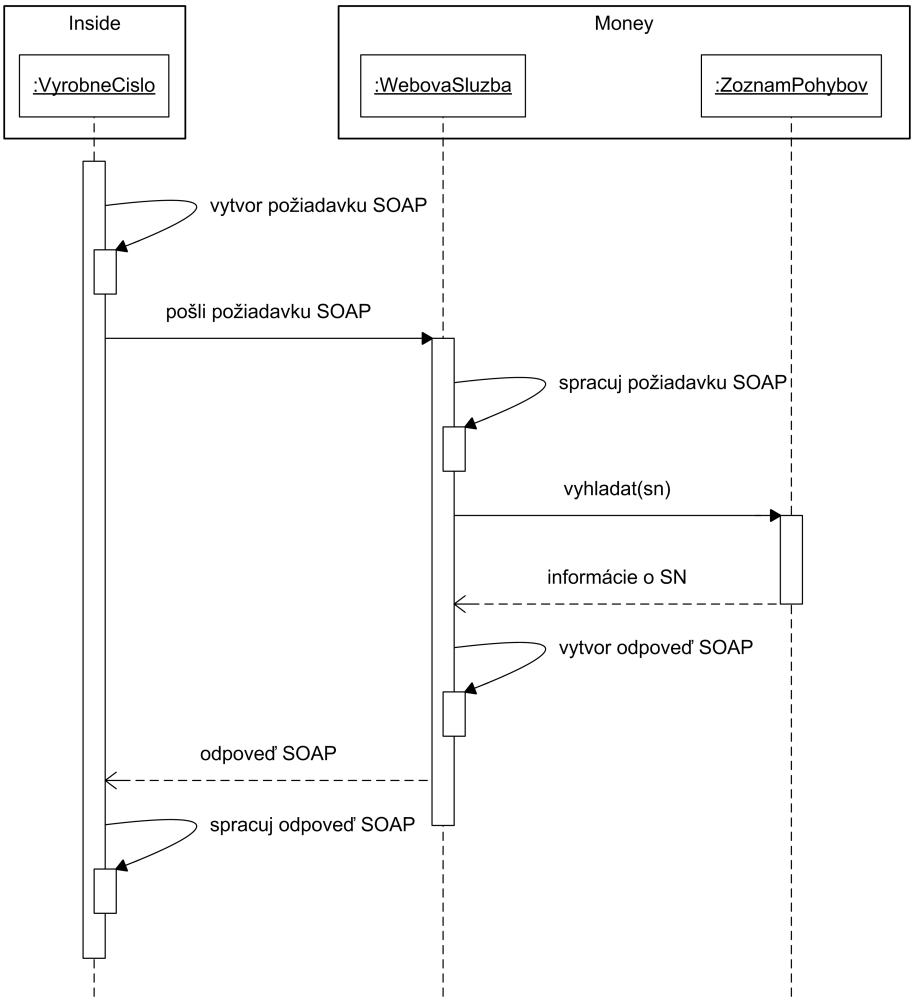


Obr. 16: Sekvenčný diagram - Pridanie objednávky

6.1.3 Vyhľadanie výrobného čísla

Na obrázku 17 je znázornený sekvenčný diagram, ktorý zobrazuje scenár prípadu užitia Vyhľadanie výrobného čísla. Potrebné je vyhľadať informácie o skladových pohybov zariadenia, podľa technikom zadaného výrobného čísla v systéme Inside. Vzájomnú interakciu začína objekt výrobné číslo v systéme Inside, ktorý vytvorí zo zadaného

výrobného čísla požiadavku SOAP a tu následne odošle objektu webovej služby v systéme Money. Objekt webovej služby požiadavku spracuje a pomocou požiadavky *vyhladaj()* na objekt zoznamu pohybov získa potrebné informácie o pohyboch. Následne vytvorí odpoveď SOAP, ktorú odošle späť objektu výrobné číslo. Ten odpoveď spracuje a zobrazí používateľovi.

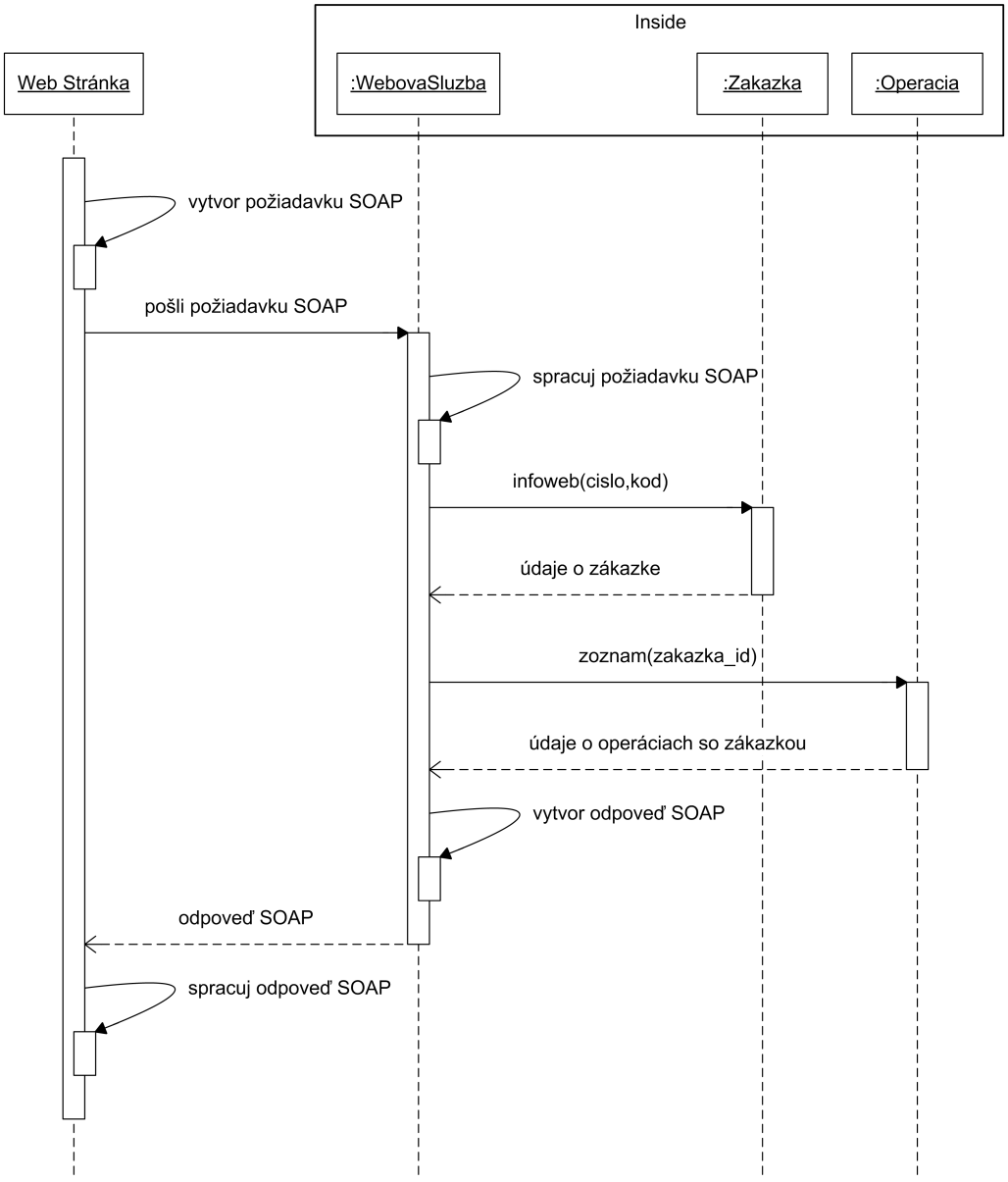


Obr. 17: Sekvenčný diagram - Vyhľadanie výrobného čísla

6.1.4 Zdieľanie informácií

Na obrázku 18 je znázornený sekvenčný diagram, ktorý zobrazuje scenár prípadu užitia Zdieľania informácií. Jedná sa o zobrazenie informácií pre zákazníka o aktuálnom stave a riešení zákazky na webovej stránke. Informácie sú vyhľadávané po zadaní čísla zákazky a prístupového kódu. Vzájomnú interakciu začína objekt webovej stránky, ktorý vytvorí

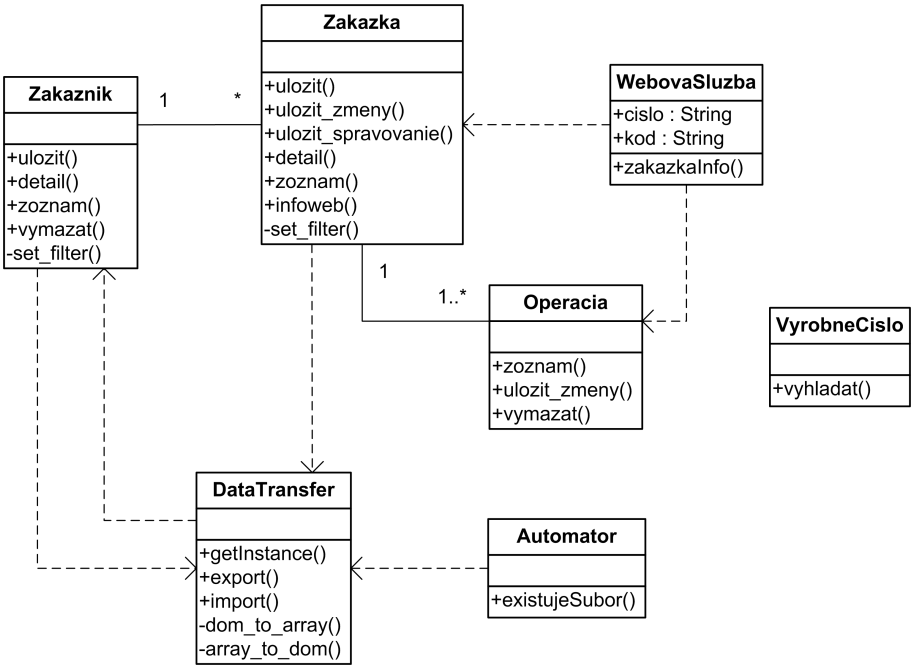
zo zadaných údajov požiadavku SOAP a tu následne odošle objektu webovej služby v systéme Inside. Objekt webovej služby požiadavku spracuje a pomocou požiadavky *infoweb()* na objekt zákazky získa potrebné informácie o zákazke. Potom požiadavkou *zoznam()* na objekt operácie získa doplňujúce informácie o zákazke. Následne vytvorí odpoveď SOAP, ktorú odošle na späť objektu webovej stránky, tá odpoveď spracuje a zobrazí zákazníkovi.



Obr. 18: Sekvenčný diagram - Zdieľanie informácií

6.2 Model tried objektov

V ďalšej fáze analýzy si spresníme statickú štruktúru rozšírenia systému. Statická štruktúra špecifikuje prvky (entity), vnútornú štruktúru týchto entít a vzťahy medzi nimi. Statickú štruktúru môžeme vyjadriť triednym diagramom, ktorý je na obrázku 19. Pri vytváraní môžeme vychádzať z modelu objektovej spolupráce, kde sme definovali základné operácie a z triedneho diagramu systému Inside, ktorý je na obrázku 3 v kapitole 2.



Obr. 19: Triedny diagram - Rozšírenie systému Inside

6.3 Dátový model

V dátovom modeli systému, ktorý je zobrazený na obrázku 4 v kapitole 2, nie je potrebné rozšírením systému nič meniť. Potrebné je len doplniť atribút *prístupového kódu* do tabuľky Zakazky, ktorý bude využívaný pri zdieľaní informácií na internetovej stránke a atribút *Money ID* do tabuľky Zakaznici, ktorý využijeme pri replikácií zákazníkových údajov. Pri pridávaní objednávky do Money zo systému Inside budeme potrebovať evidovať výrobné číslo použitého materiálu, čiže pridať atribút *výrobné číslo* do tabuľky VyuctMaterial.

7 Návrh

Cieľom etapy návrhu [4] v rámci toku činností zaoberajúcich sa analýzou a návrhom je vytvorenie modelu návrhu. Model návrhu ďalej upresňuje model analýzy vo svetle skutočného implementačného prostredia. Pojem implementačné prostredie v podstate vyjadruje možnosti použitia navrhnutých softvérových komponentov obsiahnutých v modelu analýzy na architektúru systému určeného k prevádzke aplikácie s využitím služieb už existujúcich softvérových komponentov.

Pri implementácii rozšírenia budeme naďalej využívať jazyk PHP a použitú architektúru systému Inside, opísanú v kapitole 2. Pre implementáciu webovej služby na strane systému Money, využijeme implementačné prostredie ASP.NET 2.0 s použitím programovacieho jazyka C#. Ako perzistentné úložisko dát využíva systém Money databázový systém MS SQL Server 2008.

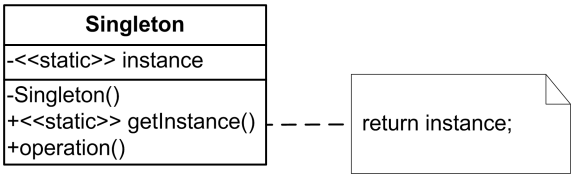
Veľké množstvo dnes používaných riešení pre komunikáciu medzi aplikáciami býva často závisle na jednej platforme, prípadne na jednej sade programovacích nástrojov. Použitím XML ako formátu správ a HTTP alebo FTP ako komunikačných protokolov je možné zaistiť kompletnú platformovú a jazykovú nezávislosť.

7.1 Použitie stávajúcej architektúry

Pri rozšírení systému Inside o úlohy, ktoré sa budú vykonávať na jeho pozadí nemusíme nijako zasahovať do používanej viacvrstvovej architektúry, ktorá je opísaná v kapitole 2. Naopak zužitkujeme jej pozitíva. Aplikačnú vrstvu (model) môžeme bez zmeny používať aj naďalej, nahradia sa len úlohy prezentačnej vrstvy (controller a view).

7.1.1 Návrhový vzor Singleton

Návrhový vzor Singleton patrí do skupiny tvoriacich vzorov. Slúži k zaisteniu existencie iba jednej inštancie danej triedy a poskytnutia globálneho prístupu k nej. Pomerne silno je ovplyvnený programovacím jazykom, v ktorom je vytváraný. Pri jeho návrhu a implementácii je nutné vychádzať z možností a obmedzení tohto jazyka. V PHP je používaný spôsob využitia statickej metódy pre vytvorenie inštancie. Konštruktor je deklarovaný ako private, čo zaistí, že nemôže byť volaný priamo, ale pre vytvorenie inštancie musí byť použitá statická metóda. Tá inštanciu ukladá do statickej private premennej, ktorú môže pri ďalšom volaní použiť.

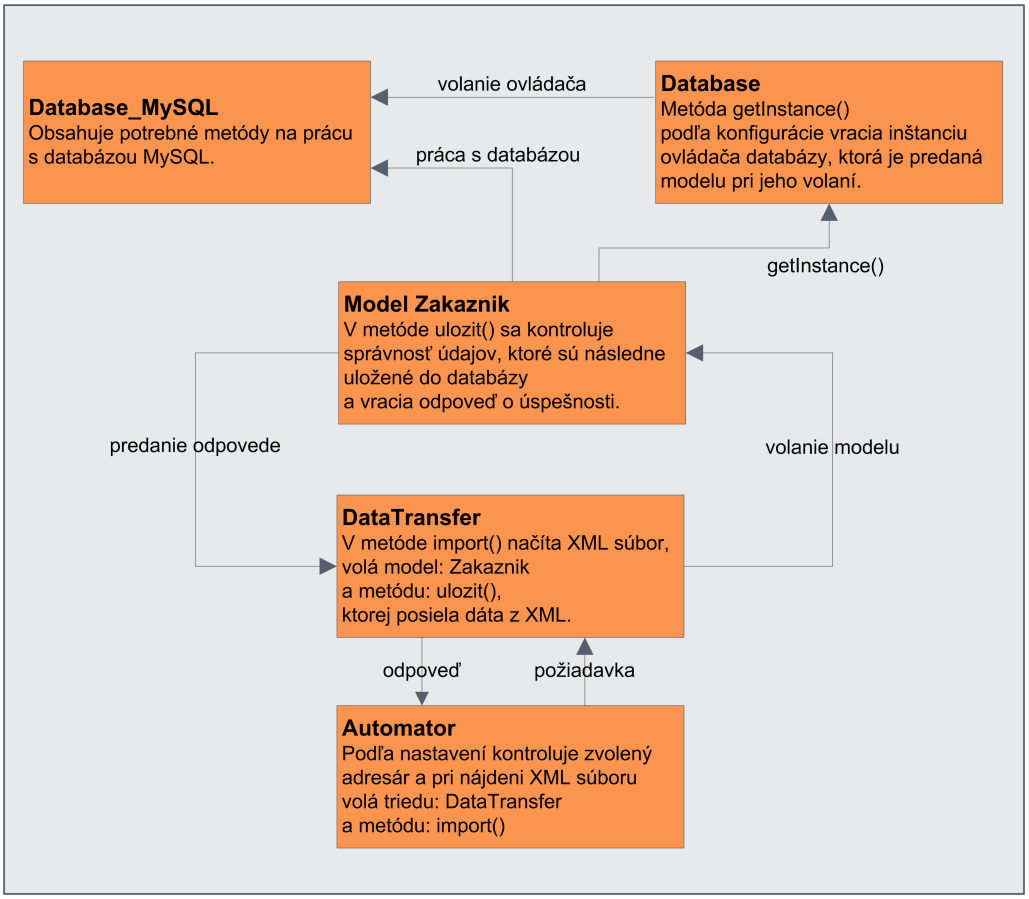


Obr. 20: Návrhový vzor Singleton

7.1.2 Kompletný návrh rozšírenia

Ak požiadavky na dáta nebudú pochádzať od klienta - používateľa, ale od iných častí aplikácie, môžeme pre danú úlohu týmito objektmi nahradiť prezentačnú vrstvu, teda v MVC komponenty controller a view. Napríklad pri importe zákazníka do systému, trieda DataTransfer zavolá model Zakaznik s inštanciou na samu seba, ako náhradu za šablónovací systém, aby mohla byť metódou *assign()* predaná odpoveď o úspešnosti alebo chybová hláška späť triede DataTransfer. K tomu okrem iného využijeme implementácie návrhového vzoru Singleton v tejto triede. Na obrázku 21 je ukážka požiadavky na import a uloženie zákazníka.

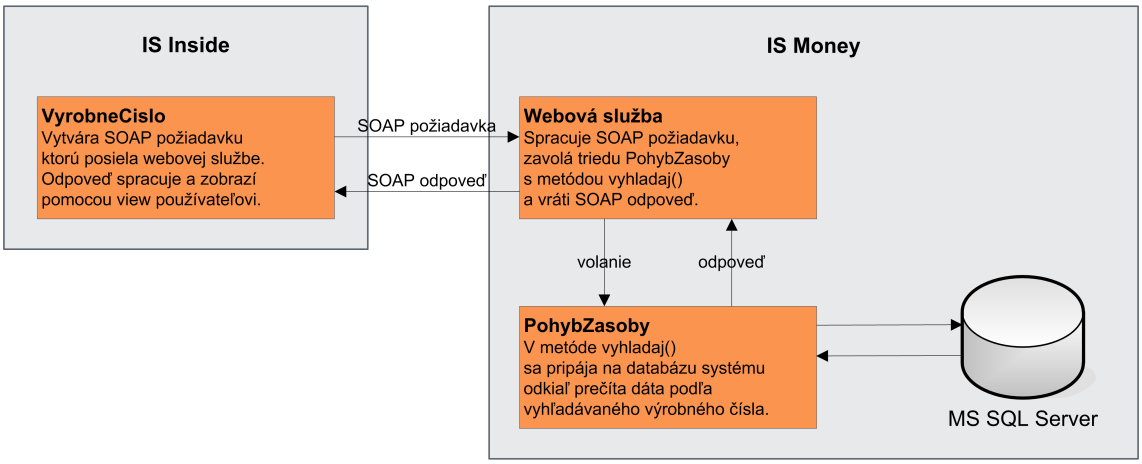
Obdobne prebieha požiadavka aj pri webovej službe, ktorá využíva model Zakazka a ten jej vracia dáta.



Obr. 21: Návrh rozšírenia - import zákazníka

Pri exporte údajov o zákazníkovi alebo zákazke do súboru XML, bude v modeli asynchrónne volaná trieda DataTransfer a jej metóda *export()*, aby uloženie súbora prebehlo na pozadí prebiehajúceho programu a prípadne zdržanie neobmedzovalo používateľa systému.

Pri tzv. vyhľadávaní výrobných čísiel v systéme Money zo systému Inside, ostane zachovaná základná architektúra systému. Implementovaný bude nový model VyrobneCislo, ktorý bude pristupovať cez webovú službu k dátam systému Money. Jeho obsluhu zabezpečí controller, prijímajúci požiadavky od používateľa. Výstup vykoná view pomocou novo vytvorenej šablóny. Na strane systému Money bude potrebné implementovať danú webovú službu, ktorá bude volať triedu PohybZasoby. Táto trieda podľa výrobného čísla vyberie z databázy systému potrebné dáta. Na obrázku 22 je ukážka požiadavky na webovú službu vytvorená na servery Money. Kvôli zjednodušeniu je vynechané zobrazenie úplnej architektúry na strane systému Inside.



Obr. 22: Návrh rozšírenia - webová služba v Money

7.2 Replikácia

Replikácia je proces kopírovania a údržby databázových objektov vo viacerých databázach. Replikované môžu byť celé databázové objekty, alebo len ich presne vymedzené časti. Replikácie sa delia na dva základne typy - synchrónne a asynchrónne.

Synchrónne replikácie garantujú v každom okamihu rovnaké dáta vo všetkých databázach replikácie. Pokiaľ by bola zaslaná požiadavka na zobrazenie dát na všetky databázy replikácie v jeden okamih, boli by výsledky zo všetkých databáz rovnaké. Dáta sa zapisujú v rámci jednej transakcie, z čoho vyplýva, že nie sú možné žiadne zmeny dát, pokiaľ sa na jednom serveri replikácie vyskytne porucha, pretože chýbajúce potvrdenie z jedného servera replikácie znemožní zápis zmien na všetkých ostatných. Synchrónne replikácie kladú veľké nároky na kvalitu pripojenia, pretože výpadok jedného servera ohrozuje funkčnosť celého systému.

Asynchrónne replikácie nezaisťujú rovnaké dáta vo všetkých databázach replikácie v každom časovom okamihu. Môže sa teda stať, že pri rovnakých dotazoch na rôzne databáze replikácie, ktoré sú zaslané v jeden okamih, budú vrátené odlišné výsledky. To je zapríčinené tým, že zmena sa vykoná najprv na jednom serveri (lokálne) a až neskôr sa

táto zmena prenáša na ďalšie servery. Replikácie nezapisujú zmeny na všetkých serveroch naraz. Asynchrónne replikácie majú nižšie nároky na kvalitu pripojenia. Dostupnosť každej repliky a jej možnosť úpravy je aj bez spojenia s ostatnými servermi.

K samotnej replikácií môžme použiť niekoľko spôsobov. Medzi známejšie patria tzv. master-slave replikácia a multi-master replikácia. Master-slave replikácia podporuje jeden hlavný server, ktorý prijíma požiadavky na zápis a čítanie dát a niekoľko podriadených serverov, ktoré umožňujú iba čítanie. Podriadený server (slave) môže mať ale možnosť požiadať hlavný server (master) o zápis zmien. Replikácia typu multi-master podporuje existenciu niekoľkých serverov s povolením zápisu a čítania dát. Teda z ktoréhokoľvek servera môže prísť požiadavka na zmenu dát a ich replikáciu na ostatné servery.

Pri asynchrónnej replikácií môže dochádzať ku viacerým konfliktom medzi dátami replík, ktoré je potrebné ošetriť, ako napríklad update confict, či delete confict. Update conflict nastáva, keď sa na dvoch synchronizovaných stranách zmenil ten istý riadok, teda riadok s rovnakým primárnym kľúčom. Delete conflict nastáva, keď na jednej strane synchronizácie bol riadok vymazaný a na strane druhej bol rovnaký riadok zmenený. Riešenie konfliktov môže byť automatické alebo manuálne. Pri automatickom riešení konfliktov sú definované pravidlá pre riešenie a tieto pravidlá rozhodnú, ktoré zmeny budú vyhodnotené ako aktuálne pri synchronizácií. Napríklad sa môže využiť časová známka, keď každý riadok má evidovaný čas poslednej zmeny. Za víťaza sa považuje strana s najneskoršou časovou známkou zmeny. Manuálne riešenie konfliktov je potrebné v prípade, že automatický systém nemá pravidlo na riešenie konfliktu. Tento konflikt je nutné zalogovať a riešiť administrátorom databázy.

7.2.1 Návrh replikácie zákazníka

Pri replikácií zákazníka medzi systémami Inside a Money použijeme multi-master asynchrónnu replikáciu, teda oba systémy menia svoje dáta lokálne a tieto zmeny budú neskôr prenesené do druhého systému. Pri riešení konfliktov medzi replikami, budeme vychádzať z tzv. optimistického prístupu, tj. nepredpokladáme, že dáta sa budú meniť v krátkom čase v oboch systémoch. Môžeme vychádzať z predpokladu, že zákazník, od ktorého sú údaje žiadané sa nachádza buď v obchodnom oddelení firmy, alebo v servisnom oddelení, nikdy nebude v oboch oddeleniach naraz.

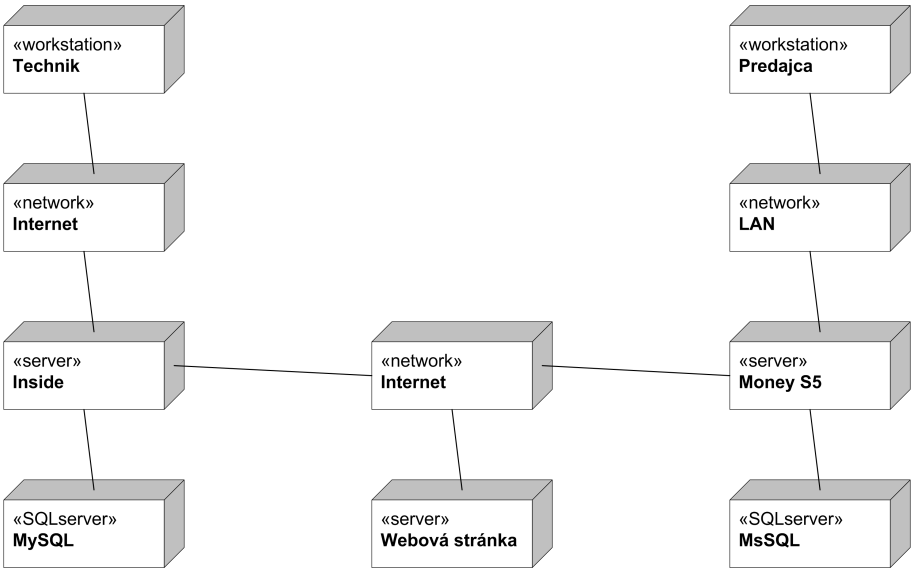
Aby sme vedeli jednoznačne identifikovať zákazníka medzi systémami musíme vzájomne preuložiť ich identifikátory (ID). Inside musí vedieť identifikátor zákazníka z Money a naopak. Keby sme toto nevedeli, nemohli by sme neskôr zákazníka upraviť, alebo vymazať v druhom systéme, pretože by sme ho nevedeli jednoznačne identifikovať. Ich preuloženie docielime tým, že po uložení nového importovaného zákazníka automaticky dáta exportujeme ale už aj s identifikátorom systému.

Samotná replikácia bude teda prebiehať následovne. Po uložení nového zákazníka v systéme Money sa automatický vyexportuje XML súbor s dátami (obsahuje Money ID zákazníka). Tento súbor bude premiestnený na server Inside, ktorý súbor prečíta a dáta importuje - uloží. Keďže sa jedná o nového zákazníka, systém Inside vyexportuje XML súbor s dátami (už obsahuje aj Inside ID zákazníka). Tento súbor bude premiestnený na

server Money, ktorý súbor prečíta a na základe svojho Money ID upraví záznam (pridá Inside ID). Všetko prebehne na pozadí bez akejkoľvek asistencie používateľa. Rýchlosť replikácie bude závisieť od rýchlosti prenosu súboru, resp. od nastavenia jej časovej pravidelnosti a taktiež od časovej pravidelnosti načítavania súborov pre import.

7.3 Model nasadenia

Zmyslom tohto modelu je špecifikovať fyzickú štruktúru prostriedkov, ktoré budú výsledný softvérový produkt prevádzkovať [4]. Na obrázku 23 je zobrazený diagram nasadenia prepojenia jednotlivých serverov. Technik pristupuje ako klient cez internet k serveru Inside, ktorý používa databázový systém MySQL. Inside je ďalej cez internet prepojený so serverom Money, ktorý používa pomocou lokálnej siete predajca. Inside je tiež cez internet prepojený s webovou stránkou firmy.



Obr. 23: Diagram nasadenia - Prepojenie systémov

8 Implementácia

Kompletný rozbor implementácie by bol mimo rozsah tejto práce, preto uvediem len zaujímavé riešenia niektorých problémov.

8.1 Prenos a čítanie súborov

Samotný prenos XML súborov medzi servermi Inside a Money prebieha pomocou protokolu FTP. Na serveri Money je spustená aplikácia určená pre automatizovanú výmenu súborov prostredníctvom FTP protokolu. V pravidelných intervaloch kontroluje vybrané adresáre alebo FTP adresáre. V nich kontroluje, či sa vyskytol nový súbor. Ak nový súbor nájde, premiestni súbor podľa nastavení do zvoleného adresára na opačnom serveri. Tento spôsob bol zvolený kvôli zabezpečeniu servera Money, kedy nie je potrebné vytvárať FTP port, cez ktorý by mohol priamo zapisovať systém Inside. Adresáre, do ktorých sú súbory premiestňované ďalej kontrolujú v pravidelných intervaloch na pozadí systémov tzv. automatory aplikácií. V Money je táto služba nazvaná *CSWAutomatic* a v rámci systému Inside bola pomenovaná *Automator*. Podobne ako nástroj na automatizovanú výmenu súborov majú za úlohu kontrolovať obsah adresára a nové súbory posúvaju ďalej na spracovanie aplikácii. CSWAutomatic volá objekt CSWImporter, ktorý importuje údaje do systému Money. Automator volá knižnicu DataTransfer a metódu *import()*, ktorá importuje údaje do systému Inside. CSWAutomatic funguje ako služba (services) systému Windows. Automator ku svojmu spúšťaniu pod PHP, ktoré beží na linuxovom serveri potrebuje správcu úloh Cron, resp. Webcron. Webcron je služba využívajúca Cron, ktorá v nastavený čas pravidelne vysiela HTTP požiadavku na zadanú adresu úlohy. Tým si môžme zabezpečiť pravidelné spúšťanie Automatoru. Vo výpise 2 je ukážka vytvorenia úlohy vo Webcron a možnosti jej nastavenia s vysvetlením.

```
*/5 * * * * http: // inside .novatech.sk/automator.php

 --------- minuta (0 - 59)
|    -------- hodina (0 - 23)
|   |    ------- den v mesiaci (1 - 31)
|   |   |    ------ mesiac (1 - 12)
|   |   |   |    ----- den v tyzdni (0 - 7)
|   |   |   |   |
*   *   *   *   *  URL

- */5 znamena pouzitie kazdych 5 minut
- jednotlive  nastavenia maju vyznam v logickom sucine
- pouzitie  hviezdicky znamena, ze na hodnote nezalezi
```

Výpis 2: Ukážka použitia a nastavenia Webcron

8.2 Transformácia XML súborov

Keďže každá aplikácia, ktorá vie importovať alebo exportovať XML súbory pracuje s vlastnou schémou XML, je potrebné po exporte alebo pred importom XML dokument transformovať na schému prijímajúceho systému. K tomuto účelu slúži XSL transformácia. XSLT definuje programovací jazyk založený na XML, ktorý slúži k prevodu XML dokumentov na iné textové formáty. Pri prepojení systémov Inside a Money sme si stanovili pravidlo, že transformácia bude prevedená ihneď po exporte. Teda exportovaný súbor XML bude v schéme akú pozná systém, ktorý bude súbor importovať. Vo výpise 3 je ukážka časti XSL súboru, používaného pre transformáciu pri exporte zákazníka zo systému Inside do systému Money. Okrem iného je v nej aj ukázané použitie podmieneného spracovania pomocou inštrukcie `<xsl:if>`, alebo zápis a čítanie atribútov elementov.

```
...
<xsl:output encoding="windows-1250" indent="yes" method="xml" />

<xsl:template match="/InsideData">
    <S5Data>
        <Firma>
            <xsl:apply-templates select="zakaznik"/>
        </Firma>
    </S5Data>
</xsl:template>

<xsl:template match="zakaznik">
    <Firma>
        <xsl:if  test="money_id!=''">
            <xsl:attribute name="ID">
                <xsl:value-of select="money_id"/>
            </xsl:attribute>
        </xsl:if>
        <Inside_ID><xsl:value-of select="@ID"/></Inside_ID>
        <Nazev><xsl:value-of select="meno"/></Nazev>
        <ICO><xsl:value-of select="ico"/></ICO>
        <DIC><xsl:value-of select="dic"/></DIC>
        <ICDPH><xsl:value-of select="icdph"/></ICDPH>
        <Adresy>
            <ObchodniAdresa>
                <Nazev><xsl:value-of select="meno"/></Nazev>
                <Ulice><xsl:value-of select="ulica"/></Ulice>
                <KodPsc><xsl:value-of select="psc"/></KodPsc>
                <Misto><xsl:value-of select="mesto"/></Misto>
            </ObchodniAdresa>
        </Adresy>
...
```

Výpis 3: Ukážka použitia XSLT

8.3 Práca s webovými službami

Volanie webových služieb v PHP zaisťuje trieda *SoapClient*, ktorej sa v konštruktore predáva adresa súboru WSDL s definíciou rozhrania a trieda sa potom o všetku komunikáciu postará sama. Ako druhý parameter konštruktora môže byť pole, ktoré nastavuje rôzne parametre ovplyvňujúce chovanie klienta. Vytvorený objekt automatický obsahuje pre všetky operácie podporované webovou službou odpovedajúce metódy. Pri zavolaní metódy sa trieda sama postará o zakódovanie dát do SOAPu, ich odoslanie webovej službe a výber hodnôt výsledku z odpovede, ktorú webová služba vráti. Vo výpise 4 je ukážka volania webovej služby v PHP.

```
try
{
    $client = new SoapClient("http://172.16.229.128/money/WebService.asmx?WSDL",
                             array('soap_version' => SOAP_1_2));

    $result = $client->Vyhladaj((object) $sn);

    // ... spracovanie $result
}
catch (SoapFault $fault)
{
    // ... spracovanie $fault->faultstring
}
```

Výpis 4: Volanie webovej služby v PHP (klient)

Vytvorenie webovej služby v PHP zaisťuje trieda *SoapServer*. Podobne ako u klienta je potrebné v konštruktore predať adresu súboru WSDL a pole s nastaveniami. Následne je potrebné implementovať triedu, ktorá pre jednotlivé operácie webovej služby poskytne metódy. Túto triedu potom stačí v objekte SoapServer zaregistrovať pomocou volania *setClass()* a obslúžiť požiadavku pomocou metódy *handle()*. Vo výpise 5 je ukážka vytvorenia webovej služby v PHP. Tá je umiestnená na serveri Inside a jej metóda *zakazkaInfo()* vracia podľa zadaného čísla zákazky informácie o zákazke.

```
$server = new SoapServer("ws.wsdl", array('soap_version' => SOAP_1_2));

$server->setClass('WebovaSluzba');

$server->handle();

class WebovaSluzba
{
    public function zakazkaInfo($Input)
    {
        $zakazka = new Zakazka();
        $info = $zakazka->infoweb($Input->cislo, $Input->kod);
```

```
            ...
        }
    }
```

Výpis 5: Vytvorenie webovej služby v PHP (server)

Vytvorenie webovej služby v .NET je oveľa jednoduchšie. Pred metódy, ktoré chceme aby sa stali operáciami webovej služby stačí pridať atribút *[WebMethod]*. Framework sa postará o všetko ostatné. Vo výpise 6 je ukážka vytvorenia webovej služby v .NET. Tá je umiestnená na serveri Money a jej metóda *Vyhladaj()* vracia podľa zadaného výrobného čísla zoznam skladových pohybov zariadenia. Túto službu využíva klient, ktorý je uvedený vo výpise 4.

```
public class WebService : System.Web.Services.WebService {

    [WebMethod]
    public List<PohybZasoby.Zoznam> Vyhladaj(string sn)
    {
        PohybZasoby ps = new PohybZasoby();
       ...
        return ps.Vyhladaj(sn);
    }
}
```

Výpis 6: Vytvorenie webovej služby v .NET (server)

Veľkou výhodou služby vytvorenej v .NET je automatické vytváranie WSDL definície, ktorá je k dispozícii pri použití adresy služby s parametrom WSDL, napríklad *WebService.asmx?WSDL*. Pri PHP musíme WSDL vytvárať ručne, prípadne vyhľadať a použiť nejaké dostupné generátory WSDL kódu, ktorých obsluha ale vyžaduje tiež potrebný čas, napríklad na definovanie typu premenných v rôznych komentárov, keďže toto jazyk PHP neumožňuje.

8.4 Asynchrónne volanie v PHP

Asynchrónne volania funkcií umožňujú vykonať kód na pozadí bežiaceho programu. Program na odpoveď nemusí čakať, alebo odpoveď spracuje neskôr. PHP priamo nepodporuje vytváranie asynchrónnych volaní. Mnohými trikmi sa ale dá docieliť tohto efektu. Jedným z riešení je použiť knižnicu CURL, ktorá je súčasťou PHP. S jej pomocou môžeme volať URL, kde máme uložený vykonateľný kód. Pri klasickom použití čaká program na vrátenie odpovede pre jej následné spracovanie. Knižnica CURL ale umožňuje nastavenie CURLOPT_TIMEOUT, čo je nastavenie určujúce dĺžku čakania na odpoveď. Ak dĺžka čakania na odpoveď prekročí nastavenú hodnotu, čakanie sa preruší

a program bude pokračovať ďalej. Vykonanie zavolaného skriptu sa ale týmto nepreruší a pokračuje na pozadí. Volanému skriptu môžeme predať dáta napríklad pomocou metódy POST HTTP požiadavky. Ukážku použitia obsahuje výpis 7. V aplikácií je tento spôsob využitý pri exportovaní súborov XML. Používateľ tým pádom môže okamžite pokračovať vo svojej práci a export dát prebieha na pozadí.

```
$ch = curl_init () ;

curl_setopt($ch, CURLOPT_URL, "http://localhost/inside/async.php");
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, $data);
curl_setopt($ch, CURLOPT_HEADER, 0);
curl_setopt($ch, CURLOPT_TIMEOUT, 1);
curl_setopt($ch, CURLOPT_FRESH_CONNECT, 1);

curl_exec($ch);
curl_close($ch);
```

Výpis 7: Pseudo-asynchrónne volanie v PHP

8.5 Singleton v PHP a jeho využitie

Ako bolo spomínané v predchádzajúcej kapitole, návrhový vzor Singleton slúži k zaisteniu existencie iba jednej inštancie danej triedy a poskytnutia globálneho prístupu k nej. Vo výpise 8 je ukážka vytvorenia Singletonu v triede DataTransfer. Pri vytváraní jej inštancie je preto potrebné využiť statickú metódu *getInstance()*. Odkaz na inštanciu je uložený v statickej private premennej *instance*. Trieda DataTransfer teda môže vlastnú inštanciu používať. To je vhodné pri volaní modelu, ktorý v konštruktore prijíma inštanciu šablónovacieho systému (view). Keďže v tomto prípade, ale výstup nebude prezentovaný používateľovi predáme modelu vlastnú inštanciu triedy DataTransfer. Ta obsahuje okrem iného aj metódu *assign()*, cez ktorú model posiela odpovede. Metóda ukladá túto odpoveď to triednej premennej a ta môže byť neskôr spracovaná. V metóde *import()* je aj ukážka volania triedy a jej metódy, ktorých názvy sú v premenných, teda vopred nie sú známe. Je to z dôvodu, že metóda *import()* je volaná Automatorom, ktorý podľa XML a nastavení určuje, ktorý model a funkcia budú zavolané.

```
class DataTransfer
{
    private static $instance;

    private function __construct() { ... }

    public static function getInstance()
    {
        if ( !isset(self::$instance) )
        {
            self::$instance = new DataTransfer();
        }
        return self::$instance;
    }

    public function import($className, $method, $xmlFile, $xslFile = null)
    {
        if (class_exists($className))
        {
            $obj = new $className(self::$instance);

            if (method_exists($obj, $method['name']))
            {
                ... citanie XML
                call_user_func_array(array($obj, $method['name']), $args)
                ... spracovanie odpovede
            }
        }
    }

    public function assign($name, $value) { ... }
}
```

Výpis 8: Singleton v PHP

8.6 Fungovanie systému

Informačný systém Inside bol rozšírený o schopnosti importu a exportu XML súborov. V rámci prepojenia s ERP informačným systémom Money boli všetky úlohy zautomatizované a prebiehajú na pozadí systémoch. Informácie umožňuje poskytovať pomocou webových služieb a rovnako vie pomocou webových služieb informácie prijímať. Požiadavky zadávateľa boli splnené a prepojenie funguje podľa predstáv firmy. Informácie o zákazníkoch sú replikované a dáta sú v oboch systémoch rovnaké. Exportovaním priebežných ale i konečných objednávok sa zvýšil prehľad o použitom materiály a znížil sa čas vybavenia zákazníka pri vyúčtovaní. Technik môže vyhľadávať informácie o skladových pohyboch zariadenia priamo zo systému Inside a to už aj pri preberaní novej reklamácie. Zákazník si môže kedykoľvek pomocou webovej stránky pozrieť aktuálny stav a spôsob riešenia jeho zákazky. Ukážky zo systémov sú súčasťou prílohy.

9 Záver

V rámci rozširovania informačného systému Inside bolo doplnených mnoho nových funkcií. Medzi viditeľné zmeny patrí aj nové používateľské prostredie, ktoré spĺňa požiadavky moderného dizajnu, ale zachováva si praktickú funkčnosť. S využitím AJAXu pomocou JavaScript knižnice jQuery sa aplikácia stala interaktívnejšou. Knižnica tiež umožnila použitie mnohých pôsobivých efektov v používateľskom rozhraní. Rozšírené boli taktiež štatistické informácie, ktoré systém poskytuje a nastavenia, ktoré umožňujú jeho ľahšiu konfigurovateľnosť. Najväčšou zmenou ale bolo rozšírenie systému z hľadiska jeho komunikácie s okolitými systémami využívanými firmou. Teda jeho prepojenie s ERP informačným systémom Money S5 a webovou stránkou firmy, čo bolo aj náplňou tejto práce.

Prepojenie so systémom Money, ktorý je využívaný vo firme ako ekonomický softvér bolo pre mňa veľkou výzvou. Potrebné bolo zachovať bezpečnosť systémov, ale riešenie muselo byť s minimálnymi úpravami znovu-použiteľné aj s inými aplikáciami. So systémom Money som pred tým nemal žiadne skúsenosti. Musel som si preto osvojiť jeho ovládanie a pochopiť spojitosti medzi dátami. Taktiež bolo potrebné odsledovať, ako firma systém využíva a používa, pretože ku niektorým výsledkom sa dá dopracovať viacerými spôsobmi. Mimo iné som absolvoval aj implementačné školenie ohľadom Money S5 od Cígler Software a tým som sa stal s firmou certifikovaným implementačným partnerom IS Money S5.

V rámci zadania si bolo potrebné ujasniť zo zadávateľom, ako prebiehajú vybrané biznis procesy vo firme a ako by mohli prebiehať efektívnejšie prepojením systémov. Následne sme spolu s manažérom firmy špecifikovali požiadavky na okolitú komunikáciu systému Inside. Na základe týchto požiadaviek sme mohli vytvoriť analýzu a návrh rozšírenia systému a jeho prepojenia. V samotnej implementácii sa použili stávajúce technológie.

Podľa požiadaviek sme ku prenosu informácií zvolili využite XML dokumentov a ku komunikácií webové služby založené na SOAP. V jazyku PHP bolo zaujímavé riešiť funkcie, ktoré sa majú vykonávať na pozadí, bez akejkoľvek interakcie používateľa, ako napríklad export a import XML. Pre export a import bolo okrem iného potrebné aj naštudovanie fungovania a použitia XSL transformácií. Veľmi prínosné pre mňa bolo tiež pochopenie fungovania webových služieb, technológii s nimi súvisiacimi a spôsob ich implementácie ako v jazyku PHP, tak aj v implementačnom prostredí ASP.NET.

Celkovo môžem zhodnotiť prácu na rozšírení informačného systému Inside ako skutočne veľmi zaujímavú a tiež veľmi prínosnú. Zoznámil som sa s novými technológiami, ale i novými riešeniami v používaných technológiách. Navrhnuté riešenia budú v budúcnosti použité aj pri iných projektoch, ako napríklad úplne prepojenie systému Money a internetového obchodu firmy. V blízkej budúcnosti je vo firme plánované nasadenie Microsoft Exchange Serveru 2010, s ktorým by mohol pomocou webových služieb aktívne komunikovať systém Inside, napríklad pri sledovaní termínov.

Vývoj informačného systému Inside sa týmto určite nezastaví. Zo strany zadávateľa prichádzajú stále nové nápady na rozšírenia. Do budúcna je plánované aj jeho poskytnutie iným firmám.

10 Literatúra

[1] Kosek J.: *PHP a XML,* Grada, Praha, 2009. ISBN 978-80-247-1116-4

[2] Skonnard A., Gudgin M.: *XML - pohotová referenční příručka,* Grada, Praha, 2006. ISBN 80-247-0972-4

[3] Holzner S.: *XSLT Příručka internetového vývojáře,* Computer Press, Praha, 2002. ISBN 80-7226-600-4

[4] Vondrák I.: *Úvod do softwarového inženýrství,* VŠB, Ostrava, 2002

[5] Vondrák I., Kožusznik J., Ochodkova E.: *Metódy specifikace softwarových systémů,* VŠB, Ostrava, 2006

[6] Vondrák I.: *Metódy byznys modelování,* VŠB, Ostrava, 2004

[7] Krátky M.: *prednášky Tvorba informačních systémů,* VŠB, Ostrava, 2007

[8] Rebroš J.: *IS firmy Novatech,* VŠB, Ostrava, 2007

[9] Grmela Z.: *Asynchronní XML schémata,* ČVÚT, Praha, 2008

[10] CZ Wikipedia: *SOAP,* dokument dostupný na URL `http://cs.wikipedia.org/wiki/SOAP`

[11] EN Wikipedia: *Web Services Description Language,* dokument dostupný na URL `http://en.wikipedia.org/wiki/Web_Services_Description_Language`

[12] Butek R.: *Which style of WSDL should I use?,* 2003, dokument dostupný na URL `http://www.ibm.com/developerworks/webservices/library/ws-whichwsdl/`

[13] Skonnard A.: *SOAP jednoducho a rýchlo,* dokument dostupný na URL `http://www.itnews.sk/tituly/infoware/free-clanky/2005-05-05/c125090-soap-jednoducho-a-rychlo`

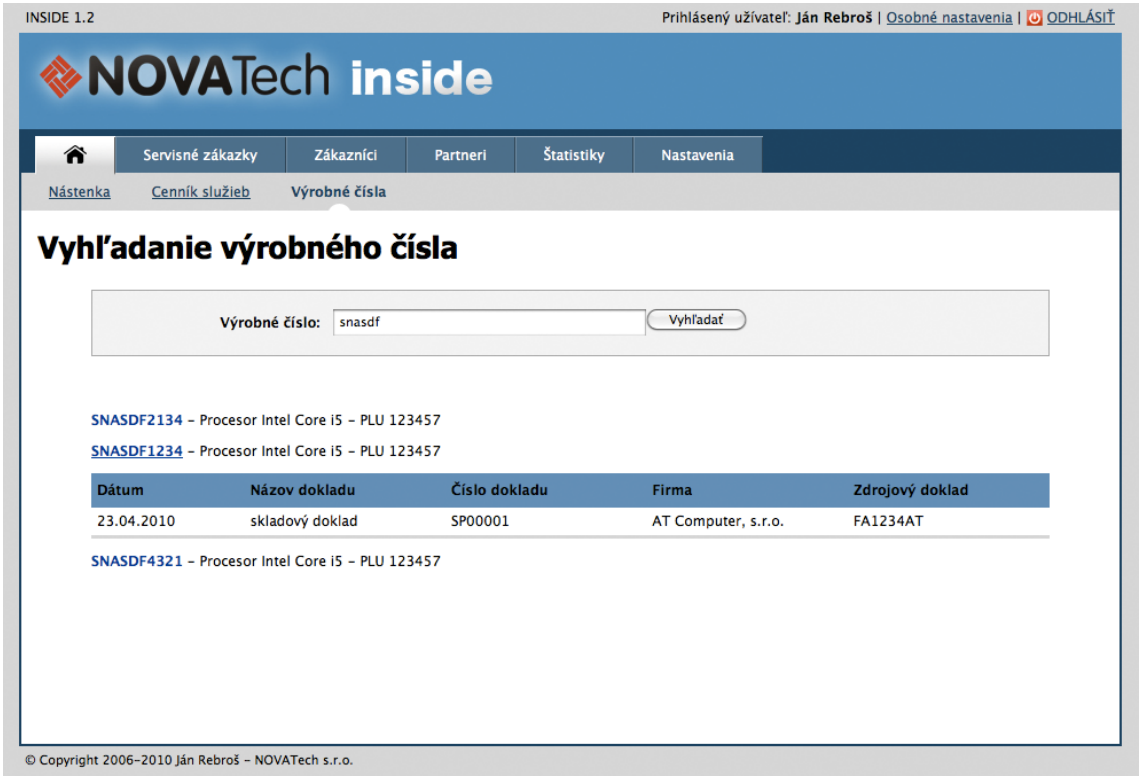
[14] Malý M.: *REST: architektura pro webové API,* 2009, dokument dostupný na URL `http://zdrojak.root.cz/clanky/rest-architektura-pro-webove-api/`

[15] Kučera F.: *Java na webovém serveru: píšeme REST API,* 2010, dokument dostupný na URL `http://zdrojak.root.cz/clanky/java-na-webovem-serveru-piseme-rest-api/`

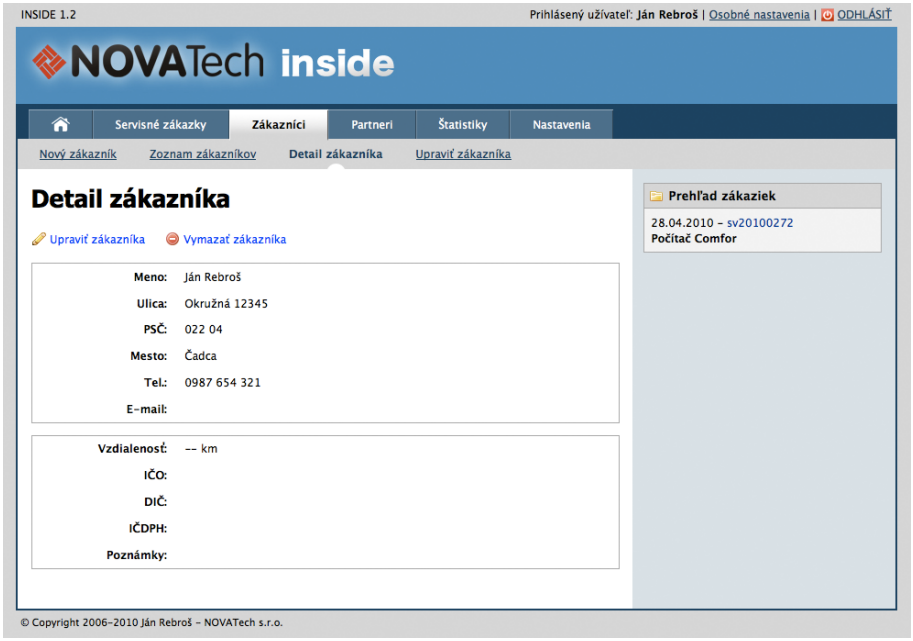
[16] CÍGLER SOFTWARE: *Money S5,* dokument dostupný na URL `http://www.money.sk/money-s5`

[17] Smarty Template Engine, `http://www.smarty.net/`

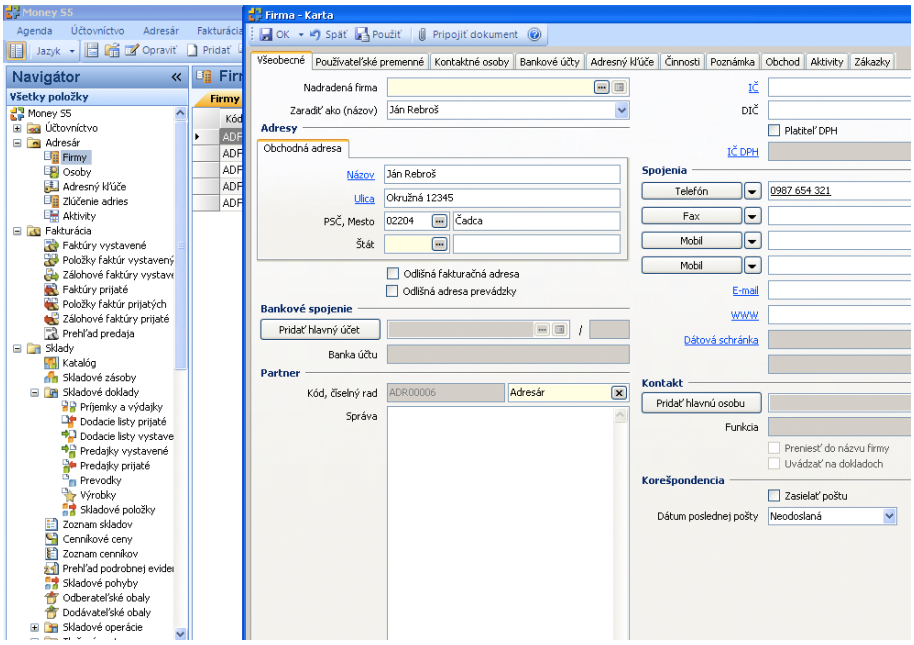
A Ukážky zo systémoch



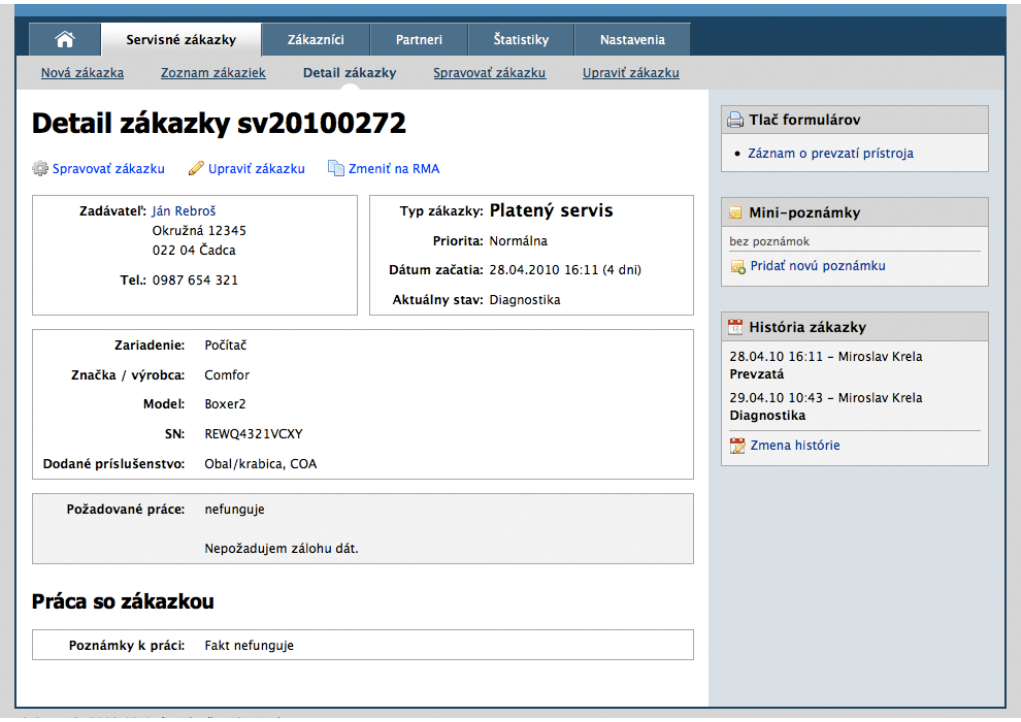
Obr. 24: IS Inside - Vyhľadanie výrobného čísla



Obr. 25: IS Inside - Detail zákazníka



Obr. 26: IS Money S5 - Adresár / Firma - karta



Obr. 27: IS Inside - Detail zákazky

Obr. 28: Webová stránka - Kontrola stavu servisnej zákazky